Computational Knowledge Discovery Techniques and Their Application to Options Market Databases

Jerome V. Healy
BSc(Hons) MSc CMath MIMA

A thesis submitted in Partial Fulfilment of the Requirements of
London Metropolitan University
for the Degree of
Doctor of Philosophy

In collaboration with the Information Technology Department
CLRC Rutherford Appleton Laboratory

October, 2004

ABSTRACT

Financial options are central to controlling investor's risk exposure. However, since 1987 parametric option pricing models have performed poorly in assessing risk levels. Also, electronic trading systems were introduced in this period, and these produce option price quotations at a rate of up to several times per second. There is a large and rapidly expanding amount of data to be analysed. A new generation of techniques for pattern recognition in large datasets has evolved, collectively termed 'computational knowledge discovery techniques' in this work. Preliminary evidence suggests that certain of these techniques are superior to parametric approaches in pricing options. Statistical confidence in models is of paramount importance in finance, hence there is a need for a systems framework for their effective deployment. In this thesis, a dedicated computational framework is developed, for the application of computational knowledge discovery techniques to options market databases. The framework incorporates practical procedures, methods, and algorithms, applicable to many different domains, to determine statistical significance and confidence for data mining models and predictions. To enable a fuller evaluation of the uncertainty of model predictions, these include a new method for estimating pointwise prediction errors, which is computationally efficient for large datasets, and robust to problems of regression and heteroskedasticity typical of options market data. A number of case study examples are used to demonstrate that computational knowledge discovery techniques can yield useful knowledge for the domain, when applied using the framework, its components, and appropriate statistical and diagnostic tests. They address an omission in the literature documenting the application of these techniques to option pricing, which reports few findings based on hypothesis testing. A contribution to the field of nonparametric density estimation is made, by an application of neural nets to the recovery of risk-neutral distributions from put option prices. The findings are also new contributions for finance. Finally, in a discussion of software implementation issues emerging technology trends are identified. Also, a case is made that future vertical data mining solutions for options market applications, should incorporate statistical analysis within the tool, and should provide access to values of partial derivatives of the models.

Acknowledgements

First and foremost, I would like to thank my supervisors, Dr. Maurice Dixon and Dr. Fang Fang Cai for their help and assistance throughout the period of my studies. I am also deeply grateful to Dr. Brian Read for his help, and for his generosity with his time. I would like to express my gratitude to Prof. Keith Jeffery for the assistance and facilities provided by the Information Technology Department of the CLRC, Rutherford Appleton Laboratory.

I am extremely grateful to Mr. Jeff Naylor, and to the Department of Computing Communications Technology and Mathematics of London Metropolitan University, without whose financial support my PhD studies would have been impossible. Many thanks to the Department Administrator, Ms. Natasha Sattaur, whose assistance ensured my dealings with the university administration ran smoothly. Thanks are also due to Prof. Ulf Eherenmark, and Dr. Tony Browne of the department for their help. I would also like to thank fellow PhD students Lonchuan Xu and Vic Page for their moral support and encouragement.

I also wish to express my gratitude to all those who helped me through many fruitful discussions. I am grateful to Dr. Gwyn Jones, for useful discussions about statistical tests. I thank Mr. Brian Eales of the Department of Economics of London Metropolitan University for helpful discussions on option pricing theory. I am grateful to Dr. Andros Gregoriou of the Department of Economics and Finance at Brunel University for his help, and for many useful discussions on finance and econometrics. I would also like to thank Prof. Christos Ionnides of the Department of Economics and Finance at Brunel University, and Ms. Sue Starkings of South Bank University for their help.

Finally, I dedicate this thesis to my parents, and thank them for their emotional support and financial help, despite difficult circumstances.

Contents

Guide to Notation 10

1 Overview 11

1.0 Introduction 12

1.1 Knowledge Discovery in Databases and Data Mining 13

1.1.1 The Opportunity 13

1.1.2 The Problem 14

1.1.3 The Solution 14

1.1.4 The Objectives of This Work 15

1.2 Financial Market Databases and Modelling 16

1.2.1 Pricing and Prediction 17

1.2.2 Other Modelling Tasks 18

1.3 A Knowledge Discovery in Databases Approach 19

1.3.1 Stages in the KDD Process 20

1.3.2 Data Selection and Preparation 21

1.3.3 Computational Knowledge Discovery Techniques 22

1.3.4 Model Training 24

1.4 The 'Data Mining Problem' and Model Reliability 25

1.4.1 The Problem of Spurious Models 26

1.4.2 Systematic Variable Selection and Avoidance of Ad-hoc Modelling 28

1.4.3 Metrics For Model Performance 30

1.4.4 Confidence in Models and Predictions 33

1.5 Major Contributions 35

1.6 Organisation of the Document 37

2 A Computational Framework for Applications to Options Market Databases **38**

2.0 Introduction 39

2.1 CRISP-DM: Origins and Structure 39

2.1.1 Methodology and Structure of CRISP-DM 40

2.1.2 KDD and the Phases of CRISP-DM 42

2.2 The CRISP-DM Reference Model (Generic Process Model) 44

2.3 Specialised Options Market Process in Terms of CRISP-DM 44

2.4 The Options Market Instantiation: Phases and Specialised Tasks 48

2.4.1 Data Selection 48

2.4.2 Data Cleaning 51

2.4.3 Data Reduction and Enrichment 53

2.4.4 Data Preparation 54

2.4.5 Data Mining 56

2.4.6 Reporting and Deployment 65

2.5 Multiple Iterations and Process Instances 68

2.6 Conclusions 70

3 Confidence in Models and Predictions **71**

3.0 Introduction 72

3.1 Motivation 72

3.2 Theory 74

3.2.1 Confidence and Prediction Intervals for Regression 74

3.2.2 Estimating Confidence and Prediction Intervals for NNs 76

3.3 Literature Review: Confidence and Prediction Intervals for NNs 78

3.3.1 Existing Methods of Estimation 78

3.3.2 Limitations of Existing Methods 81

3.4 Robust Practical Prediction Intervals and PR 82

3.4.1 Least Squares Derivation 83

3.4.2 Maximum Likelihood Derivation 85

3.4.3 Derivation of the PR Criterion 86

3.5 Practical Implementation and Tests 88

3.5.1 Testing the Method 88

3.5.2 Test Using Standard Synthetic Data 89

3.5.3 Test Using Synthetic Option Prices 93

3.5.4 Test Using Actual Option Prices 96

3.6 Conclusions 100

4 Application of the Computational Framework 101

4.0 Introduction 102

4.1 Application I: Option Pricing 102

4.1.1 Modern Parametric Option Pricing 103

4.1.2 Literature Review: Computationl Knowledge Discovery Techniques and Option Pricing 104

4.2 Pricing FTSE 100 Options 109

4.2.1 The Data: Selection, Cleaning, and other Preparation Phases 110

4.2.2 Data Mining Model Search: Pricing with Transaction Costs 112

4.2.3 NN Option Pricing Models: Architecture Selection 117

4.2.4 NNs as an Investigative Tool: Hedging With Transaction Costs 119

4.2.5 Conclusions 125

4.3 Application II: Option Implied Probability Distributions 126

4.3.1 Risk Neutral Distributions: Theory and Applications 126

4.3.2 Recovering RNDs: Existing Methods 128

4.4 Recovering RNDs for FTSE 100 Index American Put Options 131

4.4.1 Motivation 131

4.4.2 Data and Methodology 133

4.4.3 Comparison Of RNDs: European and American Put Options 134

4.4.4 Conclusions 136

5 Evaluation and Conclusions 139

5.0 Introduction 140

5.1 Evaluation of Results 140

5.1.1 The Computational Framework 140

5.1.2 Estimation of Prediction Risk, Confidence and Prediction Intervals 141

5.1.3 Results for Applications I and II 142

5.1.4 Utility of the Computational Framework 143

5.2 Implementation Considerations 144

5.2.1 Software Implementations of Individual Techniques 145

5.2.2 Data Mining Suites 145

5.2.3 Emerging Technology Trends 146

5.2.4 Statistical Analysis Within the Tool 147

5.2.5 Derivatives of Non-parametric Regression Functions 148

5.3 Conclusions 149

References 152

Glossary of Acronyms and Technical Terms 159

Appendix A: Neural Networks 164

Appendix B: Delta and Sandwich Methods 168

Appendix C: Bootstrap and Bayesian Methods 171

Appendix D: Option Pricing 175

Appendix E: Summary Statistics of RNDs 180

Appendix F: Statistical Tests 181

Appendix G: Material Published in Advance of the Thesis 185

Appendix H: Approved Program of Research 212

List of Tables

Table 1.	Classes of Computational Knowledge Discovery Techniques	23
Table 2.	Data Mining Context for Options Market Applications	45
Table 3.	GeTS Model Search Algorithm for Input Space Search	61
Table 4.	Architecture Selection Algorithm	64
Table 5.	Nested Models: True and Nominal Statistical Significance	65
Table 6.	Training Algorithm	88
Table 7.	Estimates by Proposed Network (Example#1)	91
Table 8.	Results for Example#1: Methods Compared	92
Table 9.	Results for Example#2: Synthetic Option Prices + Noise	92
Table 10.	Estimates by Proposed Network (Example#2)	94
Table 11.	Estimates by Proposed Network (Example#3)	96
Table 12.	Estimates by Proposed Network (Example#4)	97
Table 13.	Small Sample of LIFFE FTSE 100 Index Option Raw Data	111
Table 14.	Step 2: Results for OLS regression of GUM	113
Table 15.	Step 3: Results for MLP fit of GUM specification	114
Table 16.	Step 4: Variable Deletion Tests Based on Comparison of Residuals	115
Table 17.	Explanatory Power of Restricted Models for Dependent Variable	115
Table 18.	Example of Architecture Selection using Prediction Risk	118
Table 19.	Descriptive Statistics of the Dataset	122
Table 20.	Ask and Bid for Underlying Asset: Comparison of Models	122
Table 21.	Pricing with BS Using Ask and Bid Prices of Underlying Asset	123
Table 22.	Pricing with MLPs Using Ask and Bid Prices of Underlying Asset	124
Table 23.	One Month Forecast Performance: Comparison of RNDs	135
Table 24.	European v. American FTSE 100 Index Options: Comparison of RNDs.	135

List of Figures

Fig.1	The KDD Process Applied to Options Market Data.	19
Fig.2	Structure and Components of the CRISP-DM Methodology.	41
Fig.3	Correspondence of the Stages of the KDD Process to the Phases of the CRISP-DM Reference Model.	42
Fig.4	The CRISP-DM Reference Model (Generic Process Model).	43
Fig.5	Correspondence of the Phases of the CRISP-DM Reference Model to Stages of the KDD Process for Options Market Data.	46
Fig.6	Specialised Process Model for Options Market Applications in Terms of CRISP-DM.	47
Fig.7	Specialised Options Market Process: Phases and Tools.	68
Fig.8	Specialised Options Market Process: A Process Instance.	69
Fig.9	Prediction Intervals for Example#1.	90
Fig.10	Prediction Intervals for Example#3.	99
Fig.11	Prediction Intervals for Example#4.	99
Fig.12	ESX Put Option RNDs: One Month Forecast of FTSE 100 Level.	137
Fig.13	SEI Put Option RNDs: One Month Forecast of FTSE 100 Level.	137

Guide to Notation

The following symbols are used throughout the text. Other symbols are defined locally so are not included here.

$\boldsymbol{D}$	A data set
$\boldsymbol{S}$	A sample from $\boldsymbol{D}$.
$\boldsymbol{d}$, $\boldsymbol{t}$, or $\boldsymbol{y}$	Vector of targets (response, or dependent variables)
$\mu(\boldsymbol{x})$	Conditional mean of the target given the input $\boldsymbol{x}$.
$\boldsymbol{x}$	A vector of inputs (explanatory variables, or regressors)
$\boldsymbol{X}$	A matrix of inputs (explanatory variables, or regressors)
ε	A scalar random error term, or a small increment.
$\hat{\mu}(\boldsymbol{x})$ or $\mu^*(\boldsymbol{x})$	Estimated conditional mean of the target given the input $\boldsymbol{x}$.
$\boldsymbol{\beta}$	Set of parameters (coefficients) in a parametric regression function
$\boldsymbol{\Omega}$	Set of weights in a non-parametric regression function
$f(\boldsymbol{x}, \boldsymbol{\beta}), f(\boldsymbol{x}, \boldsymbol{\Omega})$	True regression functions.
$f(\boldsymbol{x}, \hat{\boldsymbol{\beta}}), f(\boldsymbol{x}, \hat{\boldsymbol{\Omega}})$	Estimated regression functions.
$E[\bullet]$	An expected value (mathematical expectation), or mean
$Var[\bullet]$	A variance
N	Sample size
$p(x)$	Probability density of the variable x.
$\hat{p}(x)$	Estimated probability density of the variable x.
$P(x)$	Probability distribution function for the variable x
O	An observed value
$\frac{d}{dx}$	Differential operator
$\frac{\partial}{\partial x}$	Differential operator (partial differentiation)
$\frac{\delta}{\delta x}$	Differential operator (functional differentiation)
$C(\bullet,\ldots,\bullet)$	Call option pricing function
$P(\bullet,\ldots,\bullet)$	Put option pricing function
$f(\boldsymbol{x})$	Function of the variable(s) x
$\sigma^2(\boldsymbol{x})$	Noise variance function
$\hat{\sigma}^2(\boldsymbol{x})$ or $\sigma^{*2}(\boldsymbol{x})$	Estimated noise variance function

CHAPTER 1

Overview

1.0 Introduction

The financial securities industry has been particularly enthusiastic in its adoption of computer systems. Over the past 15 to 20 years the industry has been transformed by the introduction of technology that captures detailed transaction information on securities trading. The London Stock Exchange abandoned 'open outcry' trading on a physical trading floor where traders met face to face as long ago as 1986. In that year, decentralised trading via computer and telephone was introduced and the trading floor closed. This was facilitated by the SEAQ and SEAQ International computer systems, which displayed stock quotes in broker's offices. In order to bring greater speed and efficiency to the market a new system, SETS (Stock Exchange Electronic Trading Service), was launched in 1997. Also the settlement service, transferring stock from seller to buyer and arranging payment, was switched to the CREST system operated by a new company CRESTCo. Between them these systems handle the stock of more than 4,700 member companies of the exchange valued at £1,200 billion. The systems daily handle approximately 140,000 transactions in more than 3,000 securities, move £30 billion in cash, and handle 3 million electronic messages. They now support more than 270,000 member accounts.

In 1998 LIFFE (London International Financial Futures Exchange), the UKs main derivatives exchange, began the process of transforming itself into an electronic market. The LIFFE CONNECT™ trading platform was introduced, allowing electronic transactions in all the exchanges products. Migration from the trading floor to LIFFE CONNECT™ began with Individual Equity Option contracts on 30 November 1998. The process was completed on 27 November 2000 with the migration of LIFFE's non-financial products, and LIFFE became a fully automated exchange. The LIFFE CONNECT™ system permits anonymous screen based trading, where traders are unaware of their actual counterparty both pre and post trade. The system is designed to handle greater order flows and transaction volumes than any other electronic trading system. It is distributed to over 400 sites in 23 countries in all major time zones, more than any other electronic trading system in the world. It offers the widest range of futures and options products offered by any electronic exchange. These comprise futures and options on short-term interest rates, government bonds, equities, indices and commodity products. Currently LIFFE CONNECT™ handles transactions worth over US$875 billion a day on average, making LIFFE the world's largest electronic

exchange by value. Following the merger of Euronext and LIFFE the Brussels and Paris derivatives markets migrated to LIFFE CONNECT™ in spring 2003, Amsterdam and Lisbon are to follow. With the conversion of the London Stock Exchange and LIFFE into electronic exchanges via the SETS/CREST and LIFFE CONNECT™ systems all UK dealing in exchange traded financial securities, and most commodity futures and options, is now screen based. These technologically driven developments place the City at the forefront of what has become an increasingly competitive marketplace.

1.1 Knowledge Discovery in Databases and Data Mining

The SETS/CREST, LIFFE CONNECT™, and similar electronic trading systems, can produce time stamped records of price quotations and transactions, up to several times per second, for all traded securities. This data is available in real time, either directly from the exchange or through data vendors such as Reuters, Bloomberg, or Bridge. Exchanges such as LIFFE also sell archived historical data generated by these systems. As a result market practitioners now have available huge historical databases to track trends, perform technical and quantitative analysis, and build and back test[1] trading models. This creates both an opportunity and a problem.

1.1.1 The Opportunity

Until the late 1980s sparse data was a dominant factor in financial modelling. Because so little data was available, models derived from data were limited in their precision, due to the resulting statistical uncertainty. Today, data availability is no longer a problem thanks to electronic trading systems, and very large high-frequency datasets[2] allow greater precision. Statistically, the more observations there are in a data set, the greater the allowable degrees of freedom, and the greater the precision of the estimators obtained. Also, discrimination between different models and model validation becomes more statistically precise. Sparse data constrains model choice to simple models with few parameters. With large datasets however, the number of independent observations approximates an asymptotic environment. This allows exploration of more complicated

[1] Unlike the 'hard' sciences (i.e. physics, chemistry, biology) it is rarely possible to set up an experiment to test a theory or model in finance. Instead models must be 'back tested' using historical data.

[2] A set of high frequency data for a single days trading in a liquid market is equivalent to 30 years of daily data [Dacorogna, M.M., Gencay, R., Muller, U., Olsen, R.B., Pictet, O.V. 2001. *An Introduction to high Frequency Finance*, Academic Press. p 6.].

(non-linear) models with more parameters. Moreover, the application of new modelling methodologies, statistical methods, and data exploration techniques becomes possible and necessary[3].

1.1.2 The Problem

The amount of data available to market practitioners today is very large and growing rapidly. This complicates investment and trading decision making, since there is so much data and complexity that may be applicable to any proposed transaction. Moreover, the instantaneous order execution and real-time data feeds of screen based dealing systems, demand rapid reaction to exploit short lived profit opportunities. Consequently, the knowledge required by decision-makers to undertake a transaction places immense stress on the analysis and modelling techniques and tools traditionally used for decision support in securities markets. When data was scarce it was natural to develop analysis and modelling techniques optimised to extract maximum information from sparse data sets. However, many of these tools and techniques are not well adapted to rapid analysis of very large complex datasets. Traditional analysis and modelling techniques usually rely on a verification-based approach. That is, the analyst hypothesises the existence of specific relationships in the data. The tools are then used to verify or reject the hypotheses. This approach rests on the prior knowledge of the analyst to formulate the hypotheses and develop the analysis based on results. The effectiveness of a verification-based approach is therefore limited by the ability of the analyst to quickly pose appropriate questions and return results, while simultaneously coping with the complexity and size of the data space.

1.1.3 The Solution

A new generation of computational techniques and tools has been developed to facilitate the extraction of useful knowledge from large complex databases. In contrast to the tools and techniques traditionally used in the financial securities domain, these analytical tools use a discovery-based approach where pattern recognition and other algorithms are applied to find important relationships in data. They can handle multiple high-dimensional data relationships concurrently, identifying exceptional or dominant relationships. These techniques are central to the fields of knowledge discovery in databases (KDD) and data mining, which aims to automate as far as possible the process

[3] Neural nets, extreme value theory, and Knowledge Discovery in Databases (KDD) for example.

of data analysis, including hypothesis selection. The techniques available include algorithms for

- CLASSIFICATION- generating a function that maps data into disparate classes.
- REGRESSION - which maps data into real-valued prediction variables[4].
- CLUSTERING - identifying a set of categories to describe data.
- DEPENDENCY MODELLING - generating models describing significant dependencies between variables.
- CHANGE and DEVIATION DETECTION - finding the most significant changes in data compared to normative or previously measured values.

They are the result of research in the fields of databases, machine learning, pattern recognition, statistics, artificial intelligence, data visualisation, and high performance computing. KDD and data mining software systems incorporate developments from all of these fields. The following standard definitions[5] are adopted here: KDD is defined as "The non trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". Data mining is defined as "A step in the KDD process consisting of particular data mining algorithms that, under some acceptable computational efficiency limitations, produce a particular enumeration of patterns".

1.1.4 The Objectives of This Work

In contrast to established techniques used in financial modelling, the computational knowledge discovery techniques considered here are relatively new. There is therefore a need to provide a systems framework for their effective deployment in the financial options domain. Statistical confidence in models is an especially important concern in finance. The objectives of this thesis are threefold. First, to outline a formal systems framework for the application of computational knowledge discovery techniques to options market databases, in the context of a KDD and data mining process consistent with established statistical principles for estimation and prediction. That is, a *statistically principled* KDD and data mining process. Second, to develop practical procedures, methods, and algorithms, to reliably estimate the true statistical significance

[4] The regression techniques traditionally used in financial econometrics are a subset of this category.
[5] Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. 1996. *"From Data Mining to Knowledge Discovery: An Overview"*, in Advances in Knowledge Discovery and Data Mining, eds. Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. AAAI Press/MIT Press, pp 1-34.

and the confidence which can be placed in predictions and models, for use within this computational framework. Thirdly, to demonstrate the procedures and methods developed, by practical application to problems of interest in the options market domain.

1.2 Financial Market Databases and Modelling

Current price quotations for stocks, bonds, and other financial instruments are available in real time from Bloomberg, Reuters, and other financial information vendors. These firms have contracts with the data originators allowing them to sell on the data. Real time quotation services are generally available only by subscription and are expensive. A dedicated terminal is often required, and these are generally closed systems, which restrict the transfer of data to other computers. Near real time quotation services are increasingly available via the Internet. These supply quotations after a time delay, they are also subscription services, but do not require dedicated terminals.

Most data driven model development in finance relies on the use of historical data however. The principal source of daily historical price data for securities is Datastream owned by Thompson Financial[6]. For some financial instruments this database is the only source of daily historical data. There is growing interest in the use of high frequency historical intra-day data (i.e. 'tick data') for financial modelling. This is increasingly available, usually directly from the data originator in CD-ROM format.

Financial market data is mostly time series data. Well-known examples are the Dow-Jones Industrial Average and FTSE 100 index series, which are frequently presented graphically in the financial media. Data consisting of a time series of several synchronously observed variables is termed 'panel data' in the terminology of finance. An example is the time series of daily closing prices of all European (American) options on the FTSE 100 index. This is an example of an 'unbalanced panel' since the number of options on offer varies from day to day. If the number of variables in the panel is constant over time it is termed a 'balanced panel'.

[6] http://www.datastream.com/

1.2.1 Pricing and Prediction

The most common tasks in financial modelling are the pricing of financial instruments in relation to other (synchronously observed) financial instruments or economic variables, and the production of predictive models for future prices or returns. In cases where the dynamics of the time series in question are well understood the modelling task reduces to the estimation of parameters of the assumed fixed functional form to fit the data. The result is called a parametric model in the terminology of finance. Provided there is sufficient data available such models can be quite accurate. However, many problems of interest in finance are not in this category.

It is often the case that a set of observations is available but the underlying data dynamics are imperfectly understood. This is the situation applying to option pricing, where most theoretical models rely on numerous simplifying assumptions to work. In these cases it is usual to assume the existence of one or more "state variables" which determine the values of the time series. A general state space form of the model can then be formally defined

$$\mathbf{y}_t = f(\mathbf{x}_t) + \varepsilon_t \tag{1.1}$$

where $\mathbf{y}_t, \mathbf{y}_{t-1}, \mathbf{y}_{t-2}, ..., \mathbf{y}_{t-(n-1)}$ is a time series of n observations, ε_t is random noise at time t, and $\mathbf{x}_t$ is a vector of state variables. Here $x_i \in \mathbf{x}_t$, and

$$x_{i,t} = g_i(\tilde{x}_{1,t}, ..., \tilde{x}_{k,t}, \mathbf{y}_t, ..., \mathbf{y}_{t-(n-1)}), \qquad i = 1, ..., n \tag{1.2}$$

where g_i is some function, and $\tilde{x}_{j,t}$ is the j^{th} element of a vector of up to k raw input variables, and t-(n-1) lagged values of the dependent variable. The multivariate form shown here contains the univariate form as a special case. The state variables may correspond to certain features of the time series. The modelling task now consists of

a) Applying selection criteria and transformations to the raw data to arrive at a suitable data representation, symbolised by the function g_i in the state space form of the model.

b) Applying an appropriate technique to fit the data, symbolised by the function f in the state space form of the model.

The problem for the modeller lies in choosing which selection criteria, transformations, and modelling technique to apply.

1.2.2 Other Modelling Tasks

Once a pricing or predictive model has been developed, a natural question is how reliable is the model? It is essential in finance to have an accurate estimate of the confidence we can place in the model, and the reliability of predictions from the model. Also, given the non-stationary nature of most financial data, there is a need to know when a model can be considered "dead", that is, no longer able to make useful predictions.

To be of practical use, models must be integrated into a trading strategy. Evaluation of a trading strategy needs to be in terms of the returns it realises, and how reliable it is in realising them, and not just statistical performance measures. This may require the definition of error measures other than the usual statistical measures of performance.

A few well-publicised large losses involving derivatives in recent years have brought the issue of financial risk, and its management, to the fore. Prudent risk management and regulatory requirements under the Basle Capital Accord, require frequent calculation of the Value at Risk (VaR) for an institution, portfolio, or market position. VaR is, formally, the probabilistic bound of market losses over a given period of time (the holding period) expressed in terms of a specified degree of certainty (the confidence interval). Put more simply, the VaR is the worst-case loss expected over the holding period within the probability defined by the confidence interval. Larger losses are possible, but with low probability. For example, a portfolio whose VaR is £20 million over a one-day holding period, with a 95% confidence interval, would have only a 5% chance of suffering an overnight loss greater than £20 million. Calculation of VaR entails modelling possible market moves over the holding period, incorporating correlations among market factors, calculating the impact of such potential market moves on portfolio positions, and combining the results to examine risk at different levels of aggregation. The three main approaches to this analysis are historical simulation, an analytical approach using a correlation matrix (as exemplified by JP Morgan's RiskMetrics™)[7], or empirical (Monte Carlo) simulation.

Options play a central role in controlling risk exposure. They make it possible to construct portfolios of assets having a selected predetermined level of risk. This is a

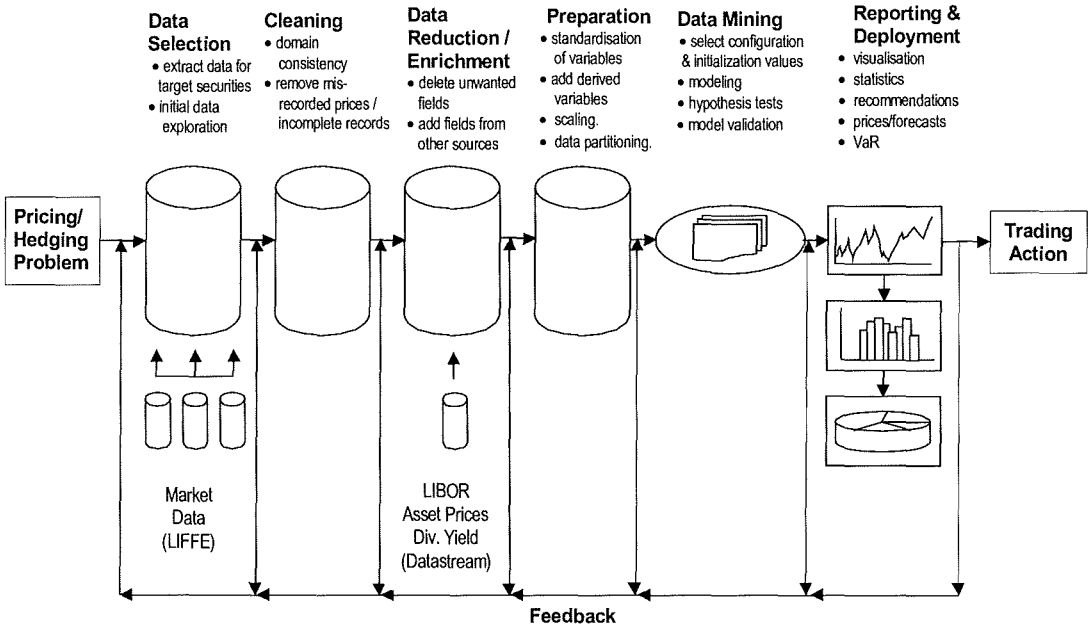
[7] J.P.Morgan/Reuters, 1996. *"Risk Metrics - Technical Document"*, Fourth Edition, New York

primary motivation for choosing options, and options market data, for this study. Moreover, options market data contains valuable information on market sentiment and risks, which can be extracted and used to aid investment and policy decisions.

1.3 A Knowledge Discovery in Databases Approach

KDD is a process not a technique. It is a multi-disciplinary activity utilising techniques developed in machine learning, statistics, database technology, expert systems, data visualisation, and other areas. The KDD approach and how it relates to financial market databases and modelling is now outlined. The process is described with reference to its application to options market databases. Fig. 1 shows an options market instantiation of the KDD process.

Fig. 1 The KDD Process Applied to Options Market Data



The KDD process is not linear. It is interactive, involving user decisions based on domain knowledge, and the possibility of looping back at any point for further iterations through one or more stages. KDD differs from traditional econometric and modelling approaches used in finance in that it is concerned with the whole process of extracting knowledge from data, in the form of patterns or models. It addresses issues of data storage and access, and the efficiency, scaling and robustness of algorithms for massive noisy datasets, in addition to questions of statistical inference common to all data analysis. KDD is especially concerned with finding understandable patterns, which constitute useful knowledge. The data mining stage of KDD relies on a variety of computational knowledge discovery techniques to find such patterns in data.

1.3.1 Stages in the KDD Process

The KDD process involves numerous stages. Fayyad et al (1996) has produced a widely accepted summary of these as follows:

1) Learning the application domain: includes relevant prior knowledge and the goals of the application.
2) Creating a target dataset: includes selecting a dataset or focusing on a subset of variables or data samples on which discovery is to be performed.
3) Data Cleaning and pre-processing: includes basic operations, such as removing noise or outliers if appropriate, collecting the necessary information to model or account for noise, deciding on strategies for handling missing data fields, and accounting for time sequence information and known changes, as well as deciding DBMS issues such as data types, schema, and mapping of missing and unknown values.
4) Data Reduction and projection: includes finding useful features to represent the data, depending on the goal of the task, and using dimensionality reduction or transformation methods to reduce the effective number of variables under consideration, or to find invariant representations for the data.
5) Choosing the function of data mining: includes deciding the purpose of the model derived by the data mining algorithm (e.g. summarisation, classification, regression and clustering)
6) Choosing the data mining algorithm(s): includes selecting method(s) to be used for searching for patterns in the data, such as deciding which models and parameters may be appropriate (e.g. models for categorical data are different to models for vectors over reals) and matching a particular data mining method with the overall criteria of the KDD process (e.g. the user may be more interested in understanding the model than in its predictive capabilities).
7) Data mining: includes searching for patterns of interest in a particular representational form or a set of such representations, including classification rules or trees, regression, clustering, sequence modelling, dependency, and line analysis.
8) Interpretation: includes interpreting the discovered patterns and possibly returning to any of the previous steps, as well as possible visualisation of the extracted patterns, removing redundant or irrelevant patterns, and translating the useful ones into terms understandable by users.

9) Using discovered knowledge: includes incorporating this knowledge into the performance system, taking actions based on the knowledge, or simply documenting it and reporting it to interested parties, as well as checking for and resolving potential conflicts with previously believed (or extracted) knowledge.

Not all of the stages enumerated above are applicable to a given KDD exercise. The application of KDD to options market data illustrated in Fig. 1 presents a simplified schema where the nine stages described above are concatenated to give six stages.

1.3.2 Data Selection and Preparation

The bulk of the literature on KDD and data mining is concerned with the data mining stage. This is understandable as the main difference between particular KDD processes lies in the computational knowledge discovery techniques used for data mining. However, successful practical implementation of KDD relies crucially on stages 1) to 6) as summarised by Fayyad et al (1996), and these usually account for 80% of the time and effort involved in specific KDD exercises. These stages are concerned with data selection and preparation prior to the data mining stage. In the example application of KDD to options market data illustrated in Fig. 1, the nine stages enumerated by Fayyad et al (1996) are replaced by the following six stages:

a) Data Selection: data for the options of interest over the chosen time period is extracted from the database of raw data (In this work we consider pricing and hedging LIFFE options on the FTSE 100 index and the database of raw data is sourced from LIFFE).
b) Cleaning: Records with inconsistent values (according to defined criteria), and important missing variables, are filtered out.
c) Data Reduction / Enrichment: Fields in the dataset not relevant to pricing and hedging the selected options are deleted. Information required to achieve the goals of the pricing or hedging exercise, obtained from other sources, is added to the dataset as new fields (e.g. appropriate LIBOR interest rates, dividend yields on the FTSE 100, FTSE100 closing prices, all obtained from Datastream).
d) Preparation: The option prices are normalised (e.g. by the exercise price) where required. The risk-free interest rate, time to maturity, implied volatility and other explanatory variables are scaled to the required input ranges. Essential derived variables are created in the dataset (e.g. moneyness, FTSE 100 price at maturity).

The data is partitioned into a training set and a test set. To ensure that each is statistically representative of the data as a whole, it is created by randomly drawing observations (without replacement).

e) Data Mining: The data-mining task is determined. For option pricing and hedging the task is regression. An appropriate computational knowledge discovery technique is selected (in this work neural nets are chosen). Initialisation criteria and network configuration are set. The network is trained and tested on the target data files. Appropriate hypothesis tests are applied. If several alternative models are created appropriate statistical tests are performed to select the best model.

f) Reporting: The results of the KDD exercise are reported. The report can include; Detailed statistics. Graphical presentations of model performance. Buy / sell / hold recommendations. Current prices. Forecast prices / returns. Hedge ratios. And VaR, or any other desired outputs. Any or all of these may be used to implement some trading action.

Stages a) to d) of the application of KDD to options market data described above are concerned with data selection and preparation and correspond to the application of the function g_i in the state space representation of the financial modelling task given in section 1.2.1. For this work, these stages were implemented systematically using relational database technology, allowing the rapid creation of target data files.

1.3.3 Computational Knowledge Discovery Techniques

Stage e) of the KDD process applied to options market data described in section 1.3.2 is the data mining stage. Data driven modelling in finance has traditionally relied on linear and non linear regression techniques including ordinary least squares regression (OLS), generalised least squares (GLS), and non linear least squares (NLLS). Recently, there has also been increasing interest in the use of non-parametric regression techniques, particularly Kernel regression. The data mining stage of the KDD process can of course employ any of the above techniques. However, data mining software suites include a variety of computational knowledge discovery techniques, and offer the financial modeller many alternative approaches. Computational knowledge discovery techniques intended for tasks other than regression though may not be suitable for modelling real valued data.

Modelling with options market data involves real valued problems. The main interest of users of this data is pricing and hedging options, and extracting risk neutral densities from quoted option prices. Hence, the applicable model is a continuous real valued function, and the appropriate data-mining task is regression. Various computational knowledge discovery techniques have been applied to the pricing and hedging of options[8]. Not all of these are regression techniques. Table 1 tabulates and classifies the main computational knowledge discovery techniques, those which have been applied to option pricing and hedging tasks are indicated with an arrow.

Table 1. Classes of Computational Knowledge Discovery Techniques

	SUPERVISED LEARNING	UNSUPERVISED LEARNING
REGRESSION	⇒ **OLS** ⇒ **NON-LINEAR LS** ⇒ **MLP** ⇒ **RBF**	
CLASSIFICATION/ CLUSTERING	⇒ ***K-NN* (Euclidean distance)** ⇒ **MLP**	• **KOHONEN NETWORKS (clustering)**
RULE BASED	• **BUILDRULE(decision tree)** ⇒ ***CART* (decision tree)** • **C.5 (decision tree)** • **GRI (association rule)**	• **APRIORI (association rule)**

In Table 1 Supervised Learning refers to techniques which are designed to fit a number of independent (explanatory) variables to one or more dependant (response) variables. Unsupervised Learning refers to techniques that are designed to search for patterns, cluster, or segment data, without reference to any response variables. Unsupervised Learning techniques are not suitable for option pricing and hedging as the data mining task is regression. Nevertheless, as Table 1 shows Supervised Learning techniques for Classification and Rule Induction, as well as for regression, have been applied to option pricing. Studies suggest[9] however, that the most suitable computational knowledge discovery technique for option pricing and hedging applications is a regression technique, namely, the form of neural net known as a multi layer perceptron (MLP). Consequently, neural nets were chosen as the specific computational knowledge

[8] See section 4.1.2 below for a literature review.

[9] Galindo, J. 1999. *"A Framework for Comparative Analysis of Statistical and machine learning Methods: An Application to the Black-Scholes Option Pricing Equation"*, in Computational Finance. (1999) Abu-Mostafa, Y.S., LeBaron, B., Lo, A.W., and Weigend, A.S. (Eds.) Proceedings of the Sixth International Conference on Computational Finance (CF99, New York, January 1999). Cambridge, MA: MIT Press. Hutchinson, J., Lo, A., and Poggio, T. 1994. *"A Non-parametric Approach to Pricing and Hedging Derivative Securities via Learning Networks."* Journal of Finance, 49, No. 3, pp 851-889.

discovery technique for use in this work. It is important to note however, that the procedures and methods developed here, apply to other computational knowledge discovery techniques of comparable flexibility.

Stage e) of the application of KDD to options market data described in section 1.3.2, corresponds here to the application of the function f in the state space representation of the financial modelling task given in section 1.2.1. For this work, the data mining stage was implemented using the commercial data mining software suite SPSS Clementine. Post processing of results from stage e), was performed in MS Excel, Statistica, Matlab, and other packages as appropriate, to obtain statistical and graphical results for stage f), the reporting stage of the KDD process described in section 1.3.2.

1.3.4 Model Training

There is an important difference between the OLS and parametric NLLS regression methods traditionally used in data driven financial modelling and econometrics, and the computational knowledge discovery techniques used for regression tasks. This difference relates to the way the minimum of the cost function is found. In OLS and parametric NLLS, the function mapping the response variable to the explanatory variables is linear in the parameters in the first case, and limited to a class of analytically tractable non-linear forms in the latter case. Indeed, for parametric NLLS the form of the non-linearity must be specified in advance. These restrictions on the parametric form a model can take mean that the error surface has a general parabolic form[10]. Hence, there is a single global minimum, and finding it is a straightforward optimisation problem. This means OLS and parametric NLLS can find the minimum of the cost function in a single pass through the data, and all the regression parameters are uniquely identified.

In contrast, neural nets and other similar computational knowledge discovery techniques for regression, do not impose restrictions on the parametric form a model can take. The function relating the response variable(s) to the explanatory variables can be arbitrarily complex, and the fit to the data arbitrarily accurate. This has several important consequences. It means that if the data are noisy the model will fit the noise unless it is

[10] Bishop C. M., 1995. *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, p15-17 and 255.

prevented, a problem termed "overfitting". It also means that the error surface is not constrained to convexity, and can in fact be highly convoluted with many peaks and local minima. As a result, for neural nets and similar techniques, finding the global minimum of the cost function requires an iterative search involving many passes through the data. Also, the weights may not be uniquely identified. For large datasets and complex models, this search may take appreciable amounts of time. However, the iterative search needs to be stopped before the data is overfitted. This is often done by stopping the search when the cost function is minimised on a separate validation dataset[11]. Performance of the model is then assessed using an independent test set. For neural nets and other computational knowledge discovery techniques, fitting data is termed "training", because of its iterative nature. It necessitates the partitioning of the available data into separate training, validation, and test sets.

The differences discussed above between regression methods traditionally used in data driven financial modelling and econometrics, and modern computational knowledge discovery techniques for regression, have resulted in differences in the ways they are used. It is not uncommon, even in published papers, to see OLS or NLLS results presented, which were obtained by fitting a single data sample, without testing of the model on an independent test set. OLS and NLLS are not sensitive to overfitting. However, they are still sensitive to sampling variation, and the best test of any model derived from data is how it performs on an unseen data set. When using neural nets and other computational knowledge discovery techniques for regression, the need to deal with overfitting dictates testing on independent data sets. The unsatisfactory procedure of presenting only in-sample results is thus avoided. In this work the statistical results presented were obtained by testing models on independent test sets.

1.4 The 'Data Mining Problem' and Model Reliability

The KDD process is intended to facilitate learning from data. Traditionally this has been the domain of statistics. The emphasis of KDD differs from that of statistics, focussing on issues of computational efficiency, scalability, and the interpretability of patterns or models extracted from data. Statistics remains a key component of KDD however. This is especially true for the data mining stage of the KDD process in a financial markets context. For options market applications, it is imperative that due

[11] These issues are discussed in greater detail in section 3.3

consideration is given to the statistical procedures used to evaluate the results of any data mining exercise. The term "data mining" has pejorative connotations in financial econometrics[12], and is used to describe indiscriminate exploration of data. This practice also termed "fishing" or "data dredging" must be avoided if meaningful results are to be obtained. The nature of this problem, how to avoid it, and the proper evaluation of models derived from a data mining process are considered in this section.

1.4.1 The Problem of Spurious Models

It is well known that if an explanatory variable is fitted to a response variable that has a trend in the same direction, a good fit can often be obtained. Even if the variables are completely unrelated, the fit may appear statistically significant. Accepting such a result is an elementary error, as it is an example of a spurious regression. Spurious regressions of this kind can be avoided by using appropriate data transformations or models. Fitting log differences of the data, or using an "error correction" model, are standard approaches to dealing with non-stationary data. In finance, it is customary to work with returns in preference to prices for this reason. Asset prices have trends, whereas returns (log price differences) are usually trend free. However, there is a subtler source of spurious models that is especially important in a data-mining context.

As an illustration, consider a database with 100 records each with 1000 fields. Suppose that each field is an independent normally distributed random sample. Thus there are 1000 possible regressors $(x_1,x_2,\ldots,x_{1000})$ all totally unrelated. Suppose this database is searched for a model by randomly selecting some field x_i as the response variable, and fitting each of the remaining regressors to it in turn using univarite OLS, to obtain 999 separate models. Suppose that the regressors are now tested for significance at the 0.05 level using a t-test. In such a test a regressor has a 1 in 20 chance of appearing significant, even if it is entirely unrelated to the response variable. In this example there are 999 regressors, so it is possible around 50 of them could appear "significant". Accepting that these "significant" regressors actually influence the response variable is a Type I error (rejecting a null hypothesis H_0 when it is true), in statistical terms. This search procedure is an extreme example of what is meant by "data dredging" or "data mining" in the econometric literature. Nevertheless, it is tempting for researchers to create multiple models in this manner, report only the "significant" ones, and discard the

[12] See e.g. Thomas, R.L. 1997. *Modern Econometrics*, Addison Wesley, p 350.

rest. In the above example, because of the multiple hypothesis tests, the nominal significance level of 0.05 is incorrect. In order to arrive at the correct significance level, the number of modelling attempts, or more precisely the number of hypothesis tests made, must be accounted for. In this example, there are n independent regressors, n mutually exclusive regressions are attempted, and n simultaneous hypotheses tested. A well-known procedure exists in statistics for multiple simultaneous hypothesis tests, according to which the true significance level for each test is[13];

$$\tilde{\alpha} = 1-(1-\alpha)^n \tag{1.3}$$

In (1.3) α is the nominal significance level (0.05 in the above example). To set the true significance level to a given value, for example 0.05 then the following equation must be solved for α to obtain the nominal significance level to be used.

$$0.05 = 1-(1-\alpha)^n \tag{1.4}$$

If in the above example, the modeller had accepted the (say) 50 nominally "significant" variables found, for inclusion in a preferred model, then the adjusted significance level for the model is given by[14];

$$\tilde{\alpha} = 1-(1-\alpha)^{(n/k)} \tag{1.5}$$

where n is the total number of regressors and k is the number included in the model. Equation (1.5) gives an adjusted significance level of 0.641 in the above example. The following approximation to (1.5) may be used for small n.

$$\tilde{\alpha} = (n/k)\alpha \tag{1.6}$$

The formulae presented above rest on the assumption that all the regressors are independent, and the correctness of the models obtained is mutually exclusive. This condition rarely applies in practice. The regressors in real world data sets are unlikely to all be independent, nor the models generated entirely mutually exclusive. If regressors are not independent the true significance level may be lower than indicated by equation (1.5). However, Lovell (1983) points out that the t-test requires an estimate of the variance of the error term, and this is likely to be underestimated in a search process like that described above. Consequently, the t-ratios are likely to be overestimated. The result is that the two errors may cancel out and, according to Lovell, equation (1.5) still provide a good indicator of the true significance level even when

[13] Thomas (1997), pp 352-353.
[14] Lovell, M.C. 1983. *"Data Mining"*, Review of Economics and Statistics, 65, pp 1-12.

regressors are correlated. Sidak (1967), Holm (1979), and others[15] have proposed improved adjustment procedures. Analogous adjustments also exist for confidence intervals. Modellers rarely obtain good results with a model on the first trial against an independent test set. They almost invariably wish to fine tune a model, add extra regressors, or compare it with other modelling techniques, in an effort to find the "best model". This all too often leads to ad-hoc explorations that make valid statistical inference impossible. Moreover, a researcher's own past results, as well as those of others, all have the potential to bias conclusions. Section 1.4.2 is concerned with a systematic approach to dealing with these difficulties.

1.4.2 Systematic Variable Selection and Avoidance of Ad-hoc Modelling

The example in section 1.4.1 shows why it is necessary to avoid ad-hoc approaches to modelling, and take account of the number of modelling attempts in a data mining model search. A systematic approach to variable selection will clearly help. But how can it be achieved? The only way of being certain a model is the "best model" is to test all possible models. An exhaustive comparison of all possible models for the data is intuitively appealing, and superficially seems a sensible way of proceeding[16]. The limitations of this approach are now considered.

Suppose there exists a dataset containing n variables which may influence some response variable y. Since each variable can be either in or out of the model the total possible number of separate models is given by;

$$m = 2^n \tag{1.7}$$

In the example in section 1.4.1 the data set contains 1000 variables. One of these is selected as the response variable, leaving 999 potential regressors. Applying equation (1.7) shows there is a total of $5.357543035 \times 10^{300}$ possible separate models. Clearly, this is an unfeasibly large number to test, even for fast computers. As few as twenty regressors still gives 1,048,576 possible models according to (1.7). Thus, an exhaustive comparison is only a feasible model search strategy where the total number of potential regressors is small.

---

[15] Sidak, Z. 1967. *"Rectangular Confidence Regions for the Means of Multivariate Normal Distributions"*, Journal of the American Statistical Association, 62, pp626-633.
Holm, S. 1979. *"A Simple Sequentially Rejective Multiple Test Procedure"*, Scandinavian Journal of Statistics, 6, pp65-70.

[16] This is the basis of simple stepwise regression.

An alternative to an exhaustive search is to allow the model search process to be guided by domain knowledge. One approach that employs this principle is the "Specific to General" modelling methodology. In the "Specific to General" methodology the modeller starts off with a simple model, perhaps including one regressor which he believes influences the response variable. The model is then tested. If it fails to satisfy the test criteria, the model may be re-estimated using alternative techniques. Extra variables or lagged values of the response variable(s) may also be added. Variations of the model are tried until a satisfactory result is finally achieved. The "Specific to General" approach is a divergent branching search without obvious stopping points[17]. The numbers of tests required and their dependencies are thus unclear, leading to a lack of control of significance levels. Consequently, it is difficult to make a meaningful judgement of the significance of the final model obtained. Hence, this approach does not adequately address the issues discussed in section 1.4.1. The "Specific to General" approach to modelling was widespread in the financial modelling domain until fifteen years ago. However, it is widely regarded as outmoded today.

A far more systematic approach to addressing the difficulties described in section 1.4.1 is provided by the "General to Specific" approach pioneered by Hendry and Richard (1983)[18]. This starts off with a general model containing all the competing models the modeller wishes to test as special cases. The special cases are said to be *nested* within the general model. There may be several levels of nesting, and some models may be non-nested on the same level. The special cases can be obtained by placing *restrictions* on the general model. The process commences by defining the general model. Once the modeller is satisfied with the specification of this, a systematic (ideally sequential) simplification search is performed, proceeding from the general model to simpler and simpler cases. At each step diagnostic tests are performed. A simpler model will not be adopted if it displays a serious deterioration in test statistics. This process is known as "testing down". In reporting a "General to Specific" model search, results for all the steps are presented. Since there are a known number of steps it is possible to estimate the true as opposed to the nominal significance level of the final model. Some uncertainty may be introduced where a choice has to be made between non-nested

[17] Hendry, D.F., and Krolzig, H.M. 2002. *"New Developments in Automatic General-to-specific Modelling"*, paper presented at the ESAM02 conference, Brisbane, July 2002, p4.
[18] Hendry, D.F., and Richard, J.F. 1983. *"The Econometric Analysis of Economic Time Series"*, International Statistical Review, 51, pp63-111.

models, or where there is more than one route to the final model. Also, choice of the initial general model may involve some preliminary data exploration. However, the approach emphasises the importance of out of sample tests for the final model, on new data not available when the model was chosen. Assuming the original general model includes all possible regressors which might influence the response variable, it is likely some irrelevant variables will be included. The error due to inclusion of irrelevant variables though, is less serious than mis-specification due to a missing relevant variable. Moreover, irrelevant variables will be removed in the simplification search.

Thus far, the model search has been considered only in terms of which regressors should be included. Computational knowledge discovery techniques such as neural nets also require initial choices on model configuration, for example the number of hidden layer nodes (degrees of freedom) the model will have. It is important to realise that two different neural net models which have the same regressors, but different configurations, count as two different models for the purposes of estimating true significance levels. Model configuration issues are discussed further in Chapter 2 section 2.4.5 of this document.

1.4.3 Metrics for Model Performance

In a simplification search for the "best model" it is desirable that the models selected at each step satisfy appropriate selection criteria. The most important criteria are

- Statistical Significance and Fit: In out of sample tests, the model predictions should not be significantly different from the actual values of the response variable(s). The coefficient of determination should explain a proportion of the variance of the response variable consistent with the statistical significance of the model predictions.
- Random Residuals: Ideally, the residuals from the model should not display any pattern. They should be free from autocorrelation and heteroskedasticity. If they are not, it suggests that the model is mis-specified. For this reason it is important to inspect a plot of the residuals, and perform diagnostic tests for autocorrelation.
- Uncorrelated Regressors and Residuals: There should be no contemporaneous correlation between the regressors and the residuals from the model. The presence of such correlations implies the model may be biased and inconsistent.

- Stable Parameters: Parameter stability over the input space is a desirable characteristic. A model with stable parameters is likely to perform well out of sample, particularly when extrapolating outside of the range of its training data.
- Feasible Predictions: A model should not produce predictions of the response variable, which are inadmissible. For example, an option pricing model which predicted negative prices.
- Parsimony: A model with fewer regressors and degrees of freedom is preferable to a more complicated model, all other things being equal. Models with fewer regressors and degrees of freedom generally perform better in out of sample tests.

Consideration of the issues discussed in section 1.4.1, employment of systematic model selection procedures as described in section 1.4.2, and testing of models using the criteria outlined above, are notably absent in the extant literature reporting applications of neural nets and other computational knowledge discovery techniques to option pricing. Indeed, it is rare for significance tests to be performed at all, much less for true significance levels to be calculated. In this literature, models are mostly evaluated by ranking them by some goodness-of-fit, or error measure. Most researchers have used some combination of the following measures[19].

Goodness-of-Fit:

a) R^2: the Coefficient of Determination. The square of the correlation coefficient ***r***. A measure of the proportion of variance of the response variable that is explained by the regressors.

b) $\overline{R}^2$: (Adjusted R^2), a) adjusted for the number of regressors in the model. Models with more regressors will generally give a better fit even if the extra regressors have no explanatory power for the response variable, b) adjusts for this effect. However, it is rarely seen in this literature.

c) The correlation coefficient ***r***: Formally, Pearson's product moment correlation coefficient. A measure of the linear association of data. A value of ± 1.0 corresponds to perfect correlation (i.e. a straight line). A value of 0.0 corresponds to no linear association, however some non-linear association may still be present.

Of these, a) is the most widely used.

[19] See e.g. Benell, J., and Sutcliffe, C. *"Black-Scholes Versus Artificial Neural Networks in Pricing FTSE 100 Options"*, *Discussion Paper 00-156*, School of Management, Southampton University, 2000

Error measures:

1) ME: The mean error, $ME = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})$, where y is the actual response variable, $\hat{y}$ is the model prediction, and n is the number of observations.

2) MAE: The mean absolute error. $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}|$

3) MSE: The mean squared error. $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2$

4) RMSE: Root mean squared error $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2}$

5) MFE: Mean fractional error $MFE = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{y})}{y_i}$

6) MAFE: Mean absolute fractional error $MAFE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(y_i - \hat{y})}{y_i}\right|$

Of these, 1) and 5) are measures of bias, whereas 2), 4), and 6) are measures of dispersion, for model predicted values. Error sizes are reported in the original units for 1), 2), and 4), the original units squared for 3), and normalised by the original units for 5) and 6). Thus 1), 2), 3), and 4) are absolute measures and can only be interpreted if the scale of the response variable is known. On the other hand 5) and 6) are scale free relative measures, but cannot be computed if any y is equal to zero.

It is arguable that statistical measures of goodness-of-fit and error performance, as defined above, do not adequately characterise the practical value of improvements in option pricing performance. The most important use of options is in hedging portfolios of securities. Also, the existence of a replicating portfolio and the absence of arbitrage are fundamental factors, which impose bounds on option prices. Thus, it has been suggested that the "tracking error" of replicating portfolios in hedging options provides a more meaningful performance measure for an option pricing model[20]. The idea is that the difference between the value of an option at expiration, and the value of its replicating portfolio of the underlying asset and a risk free bond, should serve as a practical measure of the accuracy of the model. In principle, (and assuming continuous

[20] Hutchinson, J. Lo, A. and Poggio, T. 1994. *"A Non-parametric Approach to Pricing and Hedging Derivative Securities via Learning Networks."* Journal of Finance, 49, No. 3, section3.3.

cost free hedging) this difference should be zero. In practice, it should be as small as possible. Suppose that V_t denotes the combined value of an option and its replicating portfolio, and t is some date between time zero when the option is purchased, and its maturity at time T.

$$V_t = \left(S_t \frac{\partial f_t}{\partial S_t} \right) + \left(e^{r\tau} B_{(t-\tau)} - \left(\frac{\partial f_t}{\partial S_t} - \frac{\partial f_{(t-\tau)}}{\partial S_{(t-\tau)}} \right) \right) - f_t \tag{1.8}$$

In equation (1.8) f_t is an option pricing function, S_t is the price of the underlying asset, $B_{(t-\tau)}$ is the value of a risk free bond, and τ is the interval at which the portfolio is rebalanced (usually daily). At time zero an option is sold, and a quantity of stock equivalent to the first right hand term in (1.8) is purchased. Also, a bond whose value is equal to the option plus the stock is sold. Thereafter, the portfolio is rebalanced at intervals according to equation (1.8). The tracking error is the value of (1.8) at maturity of the option, that is V_T. A performance measure based on the tracking error proposed by Hutchinson, Lo, and Poggio (1994) is

$$\xi \equiv e^{-rT} E(|V_T|) \tag{1.9}$$

Equation (1.9) is the expected value of the tracking error discounted to its present value at the risk free interest rate. Hutchinson et al (1994) also define a "prediction error" which combines the expected value and variance of the tracking error, given by

$$\eta \equiv e^{-rT} \sqrt{E^2(V_T) + Var(V_T)} \tag{1.10}$$

In equation (1.8) the option is only hedged against movement of the underlying asset, so that considerable risk remains. Alternative definitions of tracking error which take account of further risks are also available. The claim that tracking error provides a better measure of option pricing performance than statistical measures of goodness-of-fit or error measures does not appear to have been subjected to empirical tests however[21].

1.4.4 Confidence in Models and Predictions

The goodness-of-fit and error measures described in section 1.4.3 are summary measures of performance. They provide an incomplete picture of model performance, since for each of them, there is only a single global parameter for model fit or accuracy. Moreover, they reveal little about the confidence that can be placed in models and

[21] The author is unaware of any comparative empirical studies on this topic in the literature.

predictions, as they do not reveal statistical significance levels. To establish significance levels, suitable hypothesis tests (i.e. F and t-tests) should always be performed. The results of hypothesis tests can be supplemented by an appropriate choice of the goodness-of-fit and error measures given above.

However, a single global hypothesis test may not provide an adequate confidence measure in an options market context. It is not unusual for option pricing models to perform well in one part of the input space, and perform poorly in another. For example, a specific model may give poor fits at low and high exercise prices, but fit well at exercise prices close to the value of the underlying asset. The same model might also display poor fits at long maturities, or high volatilities, and good fits at short maturities and low volatilities. Consequently, market practitioners and researchers are often interested in examining model performance over various subsets of the input space. In the option pricing literature, this has traditionally been achieved by partitioning data along various dimensions. Partitioning by exercise price / moneyness and by maturity is the most frequent example. However, when data is partitioned in this manner the location of the partitions is inevitably arbitrary. As a result different values for statistical significance or other performance measures can be obtained for the same options, depending on partition size and boundary location. Clearly, this is unsatisfactory. Computing pointwise confidence and prediction intervals provides a solution to this difficulty.

With few exceptions[22], the literature documenting the application of computational knowledge discovery techniques to options market data does not make use of formal hypothesis tests or statistical significance levels for models or predictions. Goodness-of-fit and error measures alone are almost exclusively used. Pointwise confidence or prediction intervals are almost never computed, perhaps because of the difficulties involved. In contrast, the results presented in this work are based on formal hypothesis tests, and true significance levels for results are reported where appropriate. Moreover, a new practical method of computing pointwise confidence and prediction intervals, and the prediction risk criterion, for neural nets and a broad class of similar computational knowledge discovery techniques was developed and tested for this work.

---

[22] One exception is Amilon, H. 2001. *"A Neural Network Versus Black-Scholes: A Comparison of Pricing and Hedging Performances"*, Working Paper, Department of Economics, Lund University, Lund, Sweden.

1.5 Major Contributions

This section outlines the major contributions and unique features of the work presented in this thesis. These are in three main areas.

I) A detailed computational framework is presented for the application of computational knowledge discovery techniques to options market data. The framework includes the following novel features:

- The framework is designed to combine the flexibility offered by computational knowledge discovery techniques, and the methodological rigour and diagnostic techniques of established modelling approaches from econometrics and statistics, in a theoretically well founded KDD process for options market applications.
- The framework is based on a specialised Options Market Process Model described in terms of the industry standard CRISP-DM Generic Process Model.
- The framework provides practical guidance for the data preparation stages of the KDD process in an options market context, and recognises that models generated in a model selection process are associated with given "training" datasets. Complete description of a model therefore includes the metadata for the dataset used for training.
- The framework integrates a systematic approach to model search in the data mining stage of the KDD process, in terms of both variable selection and model topology (architecture), in order to address the "data mining" problem and allow estimation of the true statistical significance of resulting models.

II) To address the need to assess model predictions on a pointwise basis, a new robust practical method for obtaining confidence and prediction intervals for computational knowledge discovery techniques used for regression is developed. The method has the following novel characteristics:

- The method is applicable to neural nets and a broad class of equivalent computational knowledge discovery technique of comparable flexibility.
- The method does not use computations involving the set of network weights or parameters or its derivatives to obtain standard errors.

- The method does not require use of data resampling techniques, and is not computationally costly.
- The method is robust to diagnostic problems of regression typically encountered with options market data, for example, non-constant variance of error terms, and non-normality of residuals.
- The method allows an alternative means of estimating the 'prediction risk' performance criterion. This, and prediction intervals, can be used as model selection criteria. Thus, the method is a contribution to the field of model selection.

III) A demonstration of the computational framework is given in two practical applications. The first application is primarily concerned with the model search, and focuses on the extraction of pricing models from options market data. The second application is concerned with the discovery of patterns in options market data, namely, the implied risk-neutral distribution (RND) for the value of the underlying asset at maturity. Novel features of the practical demonstrations of the computational framework include:

- A first evaluation of the merits of neural nets for extracting option pricing models from market data in a KDD context, based on a systematic model search methodology, formal hypothesis tests, and the use of true as opposed to nominal significance levels.
- An empirical investigation of the effects of transaction costs for trading the underlying asset on the performance of option pricing models, using neural nets, the Black-Scholes model, and estimates of transaction costs for the FTSE 100 index.
- A first use of neural nets to smooth and regularise option prices in order to extract fully non-parametric estimates of RNDs from American put option price series.

Additionally, a discussion of implementation issues is presented. This focuses on the suitability of currently available software implementations of computational knowledge discovery techniques for operational deployment in options market applications. Emerging technology trends and issues requiring resolution in the next generation of software solutions are identified.

1.6 Organisation of the Document

The remainder of the thesis is organised as follows. Chapter 2 presents the computational framework. The proposed practical method for obtaining confidence and prediction intervals is described in Chapter 3. The computational framework and proposed method for obtaining confidence and prediction intervals are applied in Chapter 4 and their practical utility is demonstrated. Chapter 5 summarises and evaluates the results presented in this thesis. Software implementation issues raised by the work are discussed. Finally, conclusions are presented and directions for further research are suggested.

CHAPTER 2

A Computational Framework for Applications to Options Market Databases

2.0 Introduction

In this chapter a Computational Framework for the application of computational knowledge discovery techniques to options market data, in an overall KDD context, is outlined. The framework is intended to be prescriptive enough to ensure that models and predictions satisfy the standards of statistical rigour expected in this specialised domain, while remaining flexible enough to be easily adapted to other areas of finance. The framework was developed in terms of the emerging industry standard CRISP-DM Reference Model for data mining. However, it encompasses the full KDD process using a process model specific to the financial options market, with specialised tasks and outputs. The framework incorporates systematic approaches to data cleaning, variable selection, model topology, and statistical testing and significance. The Computational Framework and its specialised process model, can therefore be directly applied by the data analyst or software developer operating in the options market domain, in contrast to the uninstantiated CRISP-DM generic process model.

2.1 CRISP-DM: Origins and Structure

Data mining first attracted widespread market interest in the early 1990's. At that time it was still a very new immature concept. The first commercial software package for data mining SPSS Clementine, was introduced by SPSS in 1994, only ten years ago. At the time a number of large, mainly European, companies involved with data mining recognised the need for a standard process model (or framework) for data mining, if it was to be adopted by firms as a key part of their business processes. To address this need, Daimler Chrysler, SPSS, and NCR set up a consortium in 1996 which was funded by the European Commission. The consortium began work on the Cross Industry Standard Process for Data Mining (CRISP-DM) from which it takes its name. The CRISP-DM was intended to be non-proprietary, and freely available. It was also intended to be industry, tool, and application-neutral. The consortium formed the CRISP-DM Special Interest Group (SIG) as a vehicle for wider industry input in order to achieve this outcome. CRISP-DM was developed and refined in live large-scale data mining projects, over a two and a half year period. By mid 1999 when the EU funded part of the project was completed, a draft of the process model had been developed.

The completed CRISP-DM version 1.0 process model[23] was finally published in August 2000. It was quickly adopted as a de facto standard.

2.1.1 Methodology and Structure of CRISP-DM

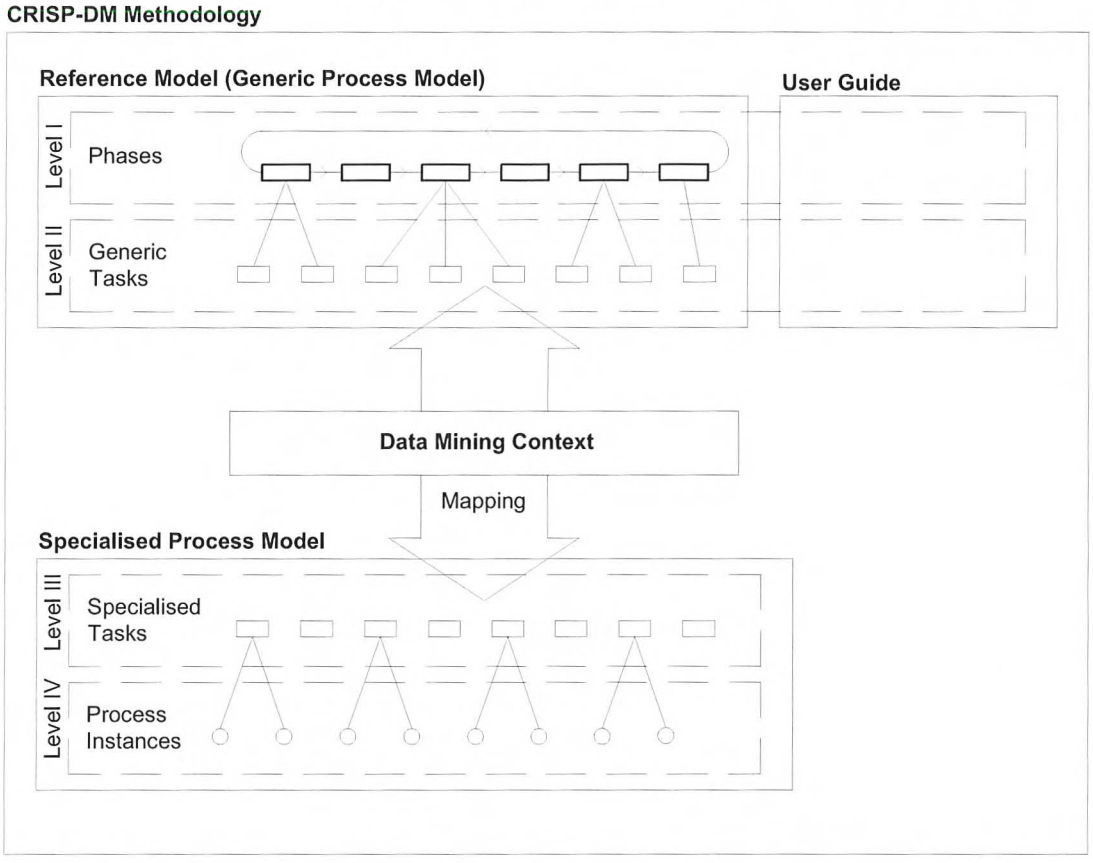
The CRISP-DM data mining methodology (or computational framework) has two components: the CRISP-DM Reference Model which is a top-down hierarchical process model of a data mining project, and the CRISP-DM User Guide which gives a more detailed explanation of the Reference Model, and guidance for its use. The CRISP-DM methodology breaks down the tasks involved in a data mining exercise into four levels, going from the general to the specific. The levels are termed: Phases, Generic Tasks, Specialised Tasks and Process Instances, respectively. The top level is the most abstract and consists of Phases, that is, the main stages involved in all KDD exercises. At the second level, each Phase is broken down into one or more Generic Tasks. The tasks are termed generic because they are intended to be industry, tool, and application-neutral, and thus general enough to apply to all possible data mining situations. The Generic Tasks are intended to be complete and stable; complete in the sense that they cover the full data mining process and all possible applications, and stable in the sense that they allow for tools and techniques yet to be developed. The third level consists of Specialised Tasks. The Specialised Tasks are specific detailed actions that are carried out for a given class of data mining exercises, in a defined domain, using specified tools and applications. The fourth level consists of Process Instances, which are records of the actions, decisions, and results, for any given single data mining exercise.

The structure of the CRISP-DM methodology is illustrated in Fig.2. The Phases and Generic Tasks are the top two levels of the CRISP-DM methodology, and make up the CRISP-DM Generic Process Model. The CRISP-DM Reference Model describes the Generic Process and thus contains *only* these levels. The Specialised Tasks, and Process Instances, are the third and fourth levels of the CRISP-DM methodology, and form the specialised levels. The Specialised Tasks, and Process Instances, are part of a Specialised Process Model for a particular industry or domain, using specific tools and applications. Before CRISP-DM can be applied in practice, the Generic Process Model (or Reference Model) must be instantiated by *mapping* the second level Generic Tasks,

[23] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R., 2000. *"CRISP-DM 1.0"*, CRISP-DM Consortium.

to third level Specialised Tasks specific to a given data mining activity, to obtain a Specialised Process Model. Two kinds of mappings between Generic and Specialised process models are defined in the CRISP-DM methodology. The first is a *single mapping*, which is a mapping for a particular data-mining project, intended for only one use. The second is where the Generic Process Model is systematically specialised according to a pre-defined Data Mining Context, for application to a given class of data mining activities. This is *a Specialised Process Model written in terms of CRISP-DM.* The Computational Framework described here, for the application of computational knowledge discovery techniques to options market data is a Specialised Process Model (explicitly developed) in terms of CRISP-DM.

Fig.2 Structure and Components of the CRISP-DM Methodology



It is important to understand that the discrete phases and tasks forming the CRISP-DM Reference Model or Specialised Process Model are not constrained to a particular sequence. In reality the process is cyclical, with only the required phases included in each cycle, terminating when the desired outcome is achieved. Thus, in practice feedback can occur from any stage to any previous stage, and given tasks may be repeated several times. In Fig.2 the loop connecting the phases in Level I indicates the cyclical nature of the process.

2.1.2 KDD and the Phases of CRISP-DM

The CRISP-DM methodology was conceived as a standard process model for data mining. However, the phases of the CRISP-DM Reference Model broadly correspond, to the stages of the KDD process as defined by Fayyad et al (1996) introduced in Chapter 1 section 1.3.1. They differ mainly in their terminology and demarcations. Thus, CRISP-DM provides a process model for the full KDD process and not merely for its data mining stage. Fig.3 shows how the stages of the KDD process as defined by Fayyad et al (1996) map on to the Phases of the CRISP-DM ver. 1.0 Reference Model.

Fig.3 Correspondence of the Stages of the KDD Process to the Phases of the CRISP-DM Reference Model.

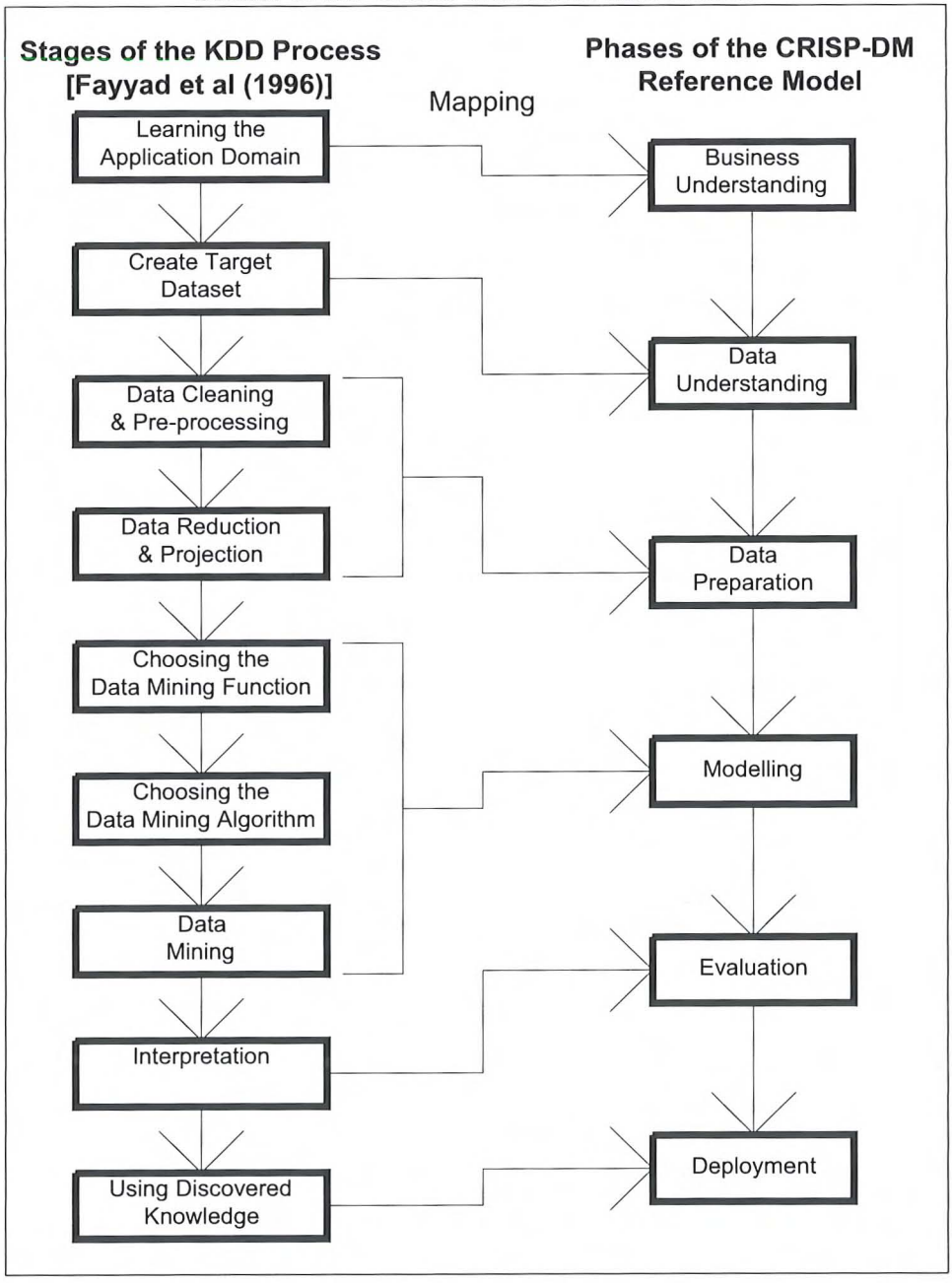
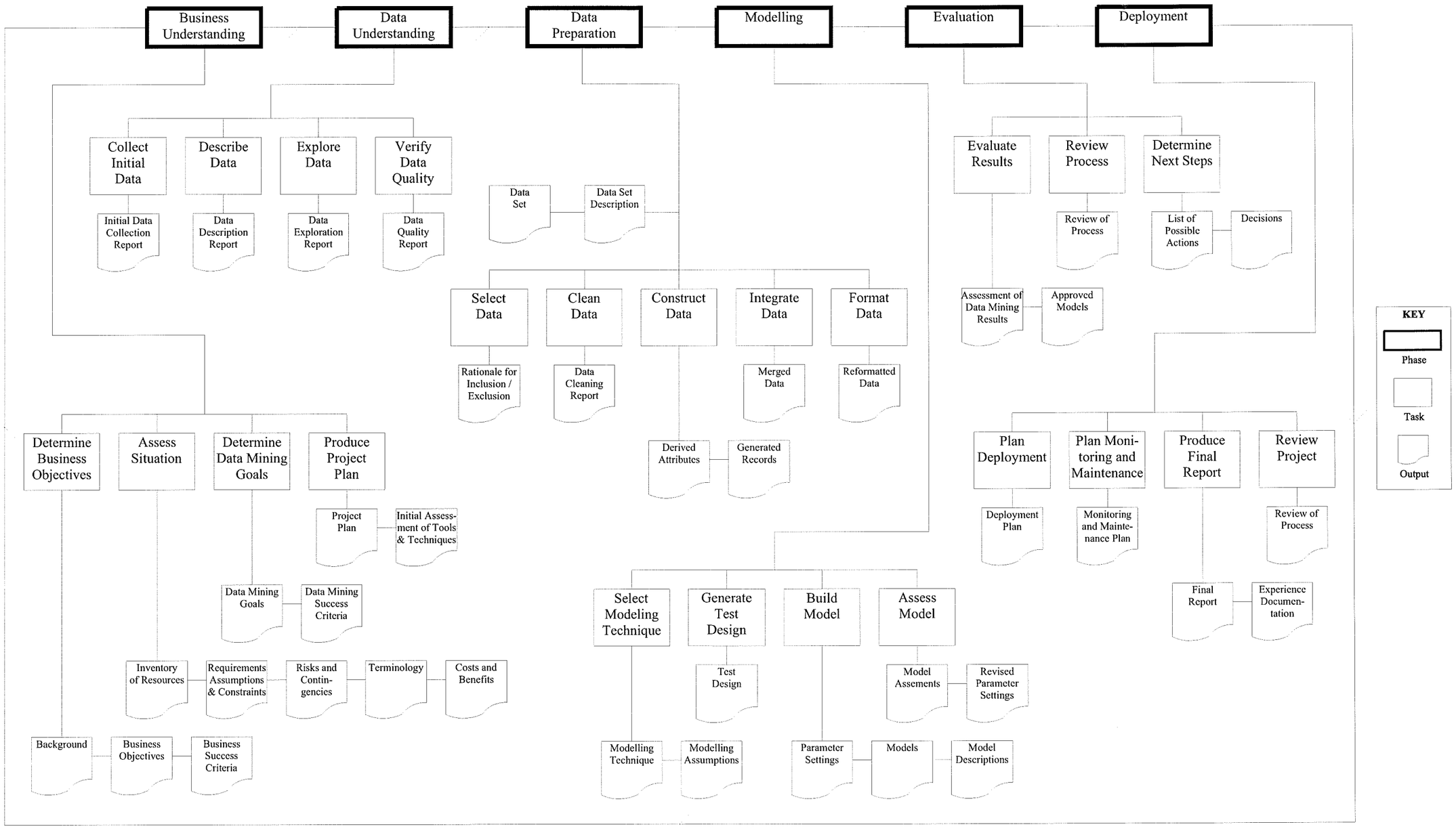


Fig.4 The CRISP-DM Reference Model (Generic Process Model).



2.2 The CRISP-DM Reference Model (Generic Process Model)

Fig.4 illustrates the full CRISP-DM Reference Model. The CRISP-DM Reference Model gives an overview of the life cycle of a data-mining project at the generic level. It describes the phases of a project, the tasks included within each phase and the relationship (sequence) of the tasks. It also describes the outputs for each task. At the Reference Model level of the CRISP-DM methodology it is not possible to identify the actual relationship (sequence) of tasks which would occur in any given data mining exercise. Rather, the Reference Model diagram illustrates an idealised sequence of phases and tasks. The arrows linking the phases and enclosing the diagram indicate the cyclical nature of the process however. The CRISP-DM Reference Model is described in detail in Section II of Chapman et al (2000).

2.3 Specialised Options Market Process in terms of CRISP-DM

Application of the CRISP-DM methodology to a specific data mining problem, or class of problems, requires the creation of a Specialised Process Model. This is done by 'instantiating' the CRISP-DM Reference Model. This section describes how the CRISP-DM Generic Process Model was systematically specialised, to provide a Specialised Process Model, or Computational Framework for Options Market Applications, developed in terms of the overall CRISP-DM methodology. Instantiating the CRISP-DM Reference Model involves mapping the CRISP-DM Generic Model to a Specialised Process Model. The mapping concerned is between the tasks of the Generic Model, and the specific tasks required in the Specialised Process Model. Essentially, a Specialised Process Model under CRISP-DM is a Generic Process Model with the generic tasks and outputs replaced by appropriate specialised tasks and outputs. Appropriate that is, to the *Data Mining Context.* The Data Mining Context is the component of the CRISP-DM methodology that drives the mapping between the generic levels (Levels I and II) of the CRISP-DM methodology and the specialised levels (Levels III and IV). The CRISP-DM Data Mining Context distinguishes between four different *dimensions* of data mining, these are :

1) **The Application Domain**

   This is the specific business or technical subject area or discipline in which the data mining takes place.

2) **The Data Mining Problem Type**

   This is the specific classification of data mining problem being dealt with.

3) **The Technical Aspect**

This describes specific issues and challenges that must be addressed during the data mining exercise(s).

4) **The Tools and Techniques**

Specifies which data mining tools (software), and techniques (computational knowledge discovery techniques), are to be employed for the data mining.

In the CRISP-DM methodology a *specific Data Mining Context* is a concrete value for one of the dimensions. The more values there are for each dimension the more concrete the Data Mining Context is. The mapping process proceeds as follows :

a) Define the Data Mining Context.

b) Remove components of the Reference Model not applicable to the specific Data Mining Context.

c) Replace generic tasks and outputs with specialised tasks and outputs relevant to the concrete characteristics of the specific Data Mining Context.

d) Rename generic components of the Reference Model to comply with the terminology and usage of the Specific Data Mining Context

Table 2 shows the specific Data Mining Context for the application of computational knowledge discovery techniques to options market data.

Table 2. Data Mining Context for Options Market Applications.

Data Mining Context				
Dimensions	**Application Domain**	**Data Mining Problem Type**	**Technical Aspects**	**Techniques and Tools**
Values	Extracting Implied Volatility.	SQL Query	Iterative search v. Newton-Raphson	VBA macro /MS Excel/Access
	Option Pricing.	Regression.	Statistical Confidence.	NN Node/SPSS Clementine
	Hedging with Options.	Regression.	Adjusting for transaction costs	NN Node/SPSS Clementine
	Extracting RNDs.	Regression.	Analytic v. numerical differentiation	NN Node/SPSS Clementine
	Value at Risk (VaR)	Regression.	Economic v. statistical VaR measures	NN Node/SPSS Clementine

Using the Data Mining Context given in Table 2, the generic phases of the CRISP-DM ver. 1.0 Reference Model are *renamed* to obtain the phases of the Specialised Process Model for Options Market Applications, in the CRISP-DM framework. The correspondence between the generic phases of the CRISP-DM ver. 1.0 Reference Model

and the stages of the specialised KDD process for options markets described in Chapter 1 section 1.3.2 illustrated in Fig.5, is the source of the new names.

Fig.5 Correspondence of the Phases of the CRISP-DM Reference Model To Stages of the KDD Process for Options Market Data

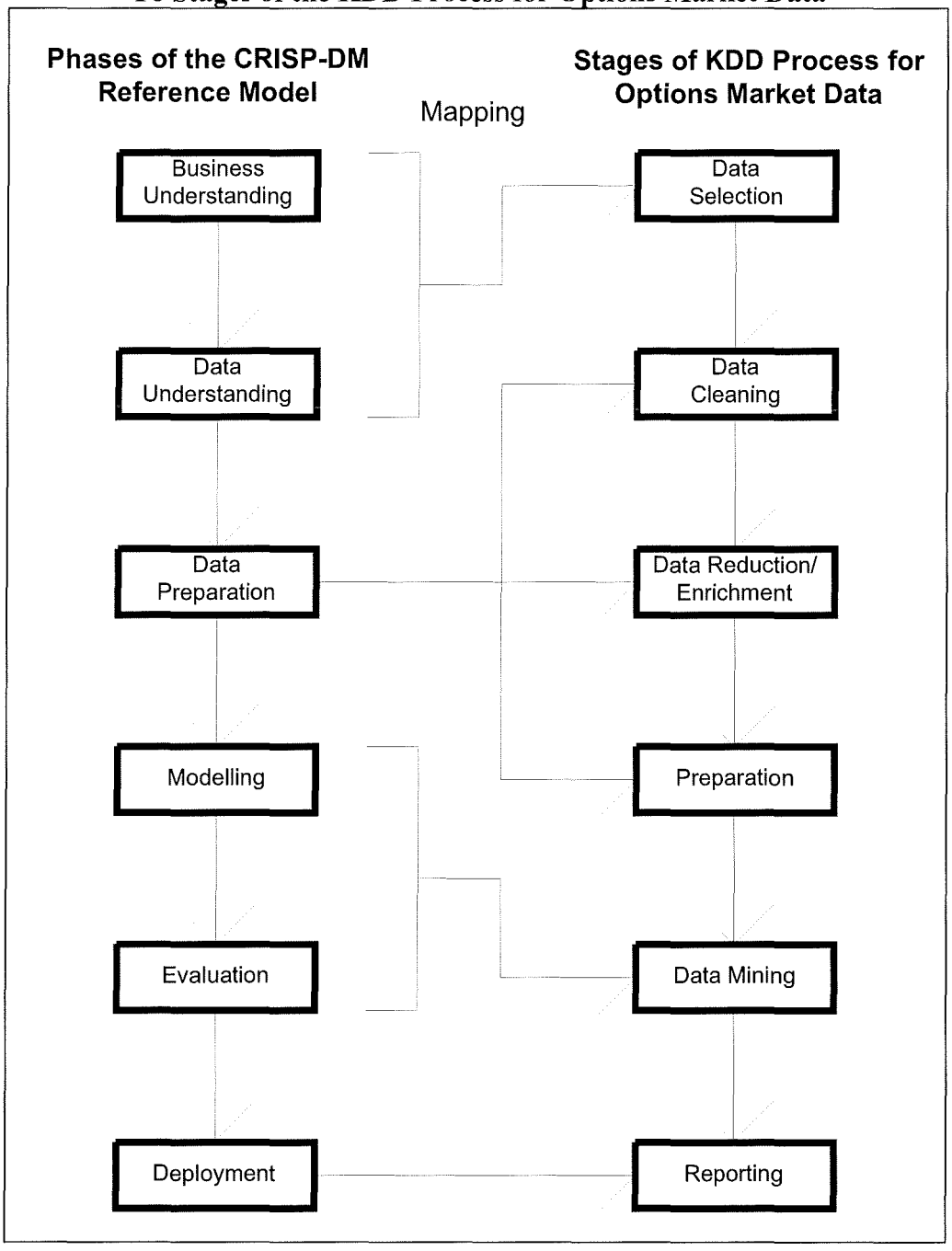
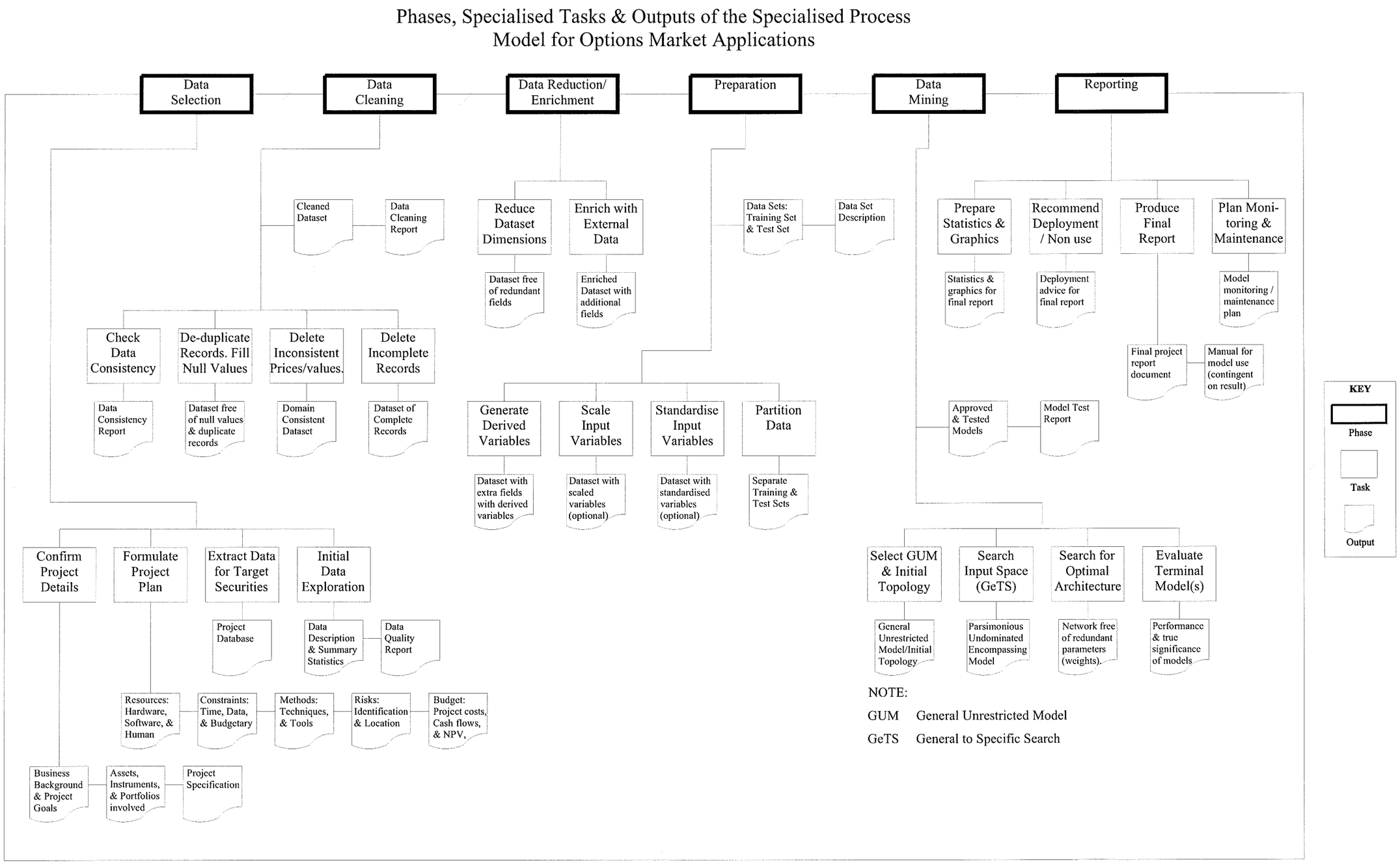


Fig.6 shows the full Specialised Process Model for Options Market Applications. This forms the Computational Framework for the application of computational knowledge discovery techniques to options market data.

Fig.6 Specialised Process Model for Options Market Applications inTerms of CRISP-DM.



2.4 The Options Market Instantiation: Phases and Specialised Tasks

The previous section described the use of the CRISP-DM methodology to create a Specialised Process Model for Options Market Applications. In this section the specific Specialised Tasks and outputs of the Specialised Process Model are elaborated and detailed. Domain knowledge helps suggest which specialised tasks should be included in any specialised process model. However, extensive practical experience of using computational knowledge discovery techniques for option pricing in the course of this research was an important factor in guiding this choice. The application domain is well defined in this case, and the available data is in a format specified by the originating institutions. Thus, the Business Understanding and Data Understanding phases of the CRISP-DM generic process simplify to the choice of data to use. The first phase in the Specialised Process Model for Options Market Applications is therefore Data Selection.

2.4.1 Data Selection

In an options market context a practitioner may need to price options for one or more of the following reasons :

a) The practitioner needs to know whether the quoted prices for a proposed purchase or sale of options is reasonable.

b) The practitioner wishes to hedge a portfolio of securities and needs a suitable pricing model from which to derive the hedge ratios.

c) The practitioner wishes to extract risk neutral densities from market prices of options for forecasting or risk management purposes, and requires a suitable. pricing model for smoothing purposes.

d) The practitioner wishes to price new or non-tradable securities by using a replicating portfolio of traded options.

d) The practitioner wishes to price options for the purposes of calculating the Value at Risk (VaR) of a replicating portfolio of securities.

The nature of the problem, the specific securities involved, and domain knowledge, will dictate the selection of data involved. The data miner will need to select the appropriate data from the historical database (data warehouse) maintained by the client organisation or obtain it from the originating institution.

Task: Confirm Project Details

The first specific task in the Data Selection phase is confirmation of the details, scope, goals, and business purpose of the proposed data mining / modelling exercise with the client. Information the data miner requires includes : The specific securities involved. The client's goals for the data mining exercise. The models or deliverables the client requires. How the client proposes to deploy any models developed. Relevant details of any related investment or hedging programme. The time, resources, and budget, available to complete the data mining exercise. It is imperative that all pertinent information is available before proceeding to the next task.

Output: Project Specification

A document giving the business background and all relevant particulars of the proposed data mining exercise. This will be a contractual document, specifying the nature and goals of the project to be undertaken, and serving as instructions for the data mining team. The Project Specification can range in size from a single A4 page to a large document depending on the complexity of the data-mining task to be undertaken.

Task: Formulate Project Plan

When the client has signed the Project Specification and the data mining team has received instructions to proceed, project planning can take place. A detailed Project Plan, developed using suitable formal methods for project management and control is required. Formal project control methods which may be relevant here include PRINCE and DSDM[24]. However, organisations may have their own preferred method. The plan should include schedules, critical paths, and milestones. Responsibilities should be clearly allocated between the members of the project team. The plan should state the resources, constraints, methods, risks, and budget applying to the project. The completed plan should be agreed and signed off by the client.

Outputs:

Resources

The Project Plan should list the hardware, software, and human, resources that will be required to attain the project goals. These may differ from, or exceed, the resources the

[24] Bentley, C. 2002. *'PRINCE2: A Practical Handbook'*, Butterworth-Heinemann, 2 edn., ISBN: 0 7506 53302.
Stapleton, J. 1997. *Dynamic Systems Development: The Method in Practice*, Addison-Wesley, Longman, Harlow, England, ISBN 0-201-17889-3.

client initially allocated to the project. In that case, the client will need to be informed, and agreement reached on the actual resources to be used. The client needs to be aware at the outset of any constraints on attaining project goals arising from lack of required resources.

Constraints

The time, data, and budgetary constraints, affecting attainment of the project goals should be identified and allowed for in the project plan.

Methods

The plan should state which computational knowledge discovery technique(s) are to be used for the project, and why. It should also specify what tools (software) are to be used to implement the techniques.

Risks

The project plan should identify any risk factors that may prevent attainment of the project goals, quantifying the risk wherever possible. The location in the project plan where (when) these risks occur should be specified.

Budget

If the project plan calls for a large scale, ongoing data mining project, substantial operational costs will be incurred, and cost of capital and cash flow considerations will come into play. In this situation the project should be evaluated using discounted cash flow techniques, and the NPV used as a project acceptance criteria.

Task: Extract Data for Target Securities

Once the project has been specified and a detailed project plan has been drawn up the data miner is in a position to extract data for the specific options involved. The required data may be obtained either from the client's own historical database (data warehouse) of options market data, or downloaded (on payment of a subscription) from the relevant financial institution. The nature of the data-mining project will dictate the dates and recency of the option prices involved, and the overall size of the data set. The data miner will need to import the raw data files into the software application to be used for the initial data exploration. This is often a non-trivial task involving complex file format conversions.

Output: Project Database

The output from this task is a project dataset of raw data in a format suitable for further processing using the tools chosen by the data miner. Relational database management systems (RDBMS) for example, are suitable tools for this stage of the process.

Task: Initial Data Exploration

At this stage an initial data exploration should be carried out to gain familiarity with the project dataset and to check for any obvious anomalies. The initial data exploration can be as simple as calculating summary statistics and plotting variables of interest.

Outputs:

Data Description and Summary Statistics

The data description is metadata describing the source of the project dataset in sufficient detail to create a clear audit trail. The summary statistics will be required for the Data Cleaning phase.

Data Quality Report

The data quality report is a brief statement concerning the quality of the raw data in the project database, confirming whether it is suitable for the purposes of the project. It may take the form of ticking boxes in a checklist.

2.4.2 Data Cleaning

Data originating from organised exchanges where options are traded, such as LIFFE, is normally of high quality. Even so, it almost invariably contains some bad quotes and other errors. These errors are of two types :

- Human Errors

  These can be either unintentional such as typing errors, or intentional errors such as dummy quotes generated for technical testing.

- System Errors

  These are errors arising from computer system failures or flaws.

Mining data containing errors will produce poor results. If there are extreme outliers present in the data the results may well be unusable. Even without errors, the raw data may be unusable due to the existence of null values, that is, fields which have been left blank. Data mining software will not normally accept null values as inputs, and it is necessary to eliminate them by applying a zero fill or other coding scheme. Relational database management systems (RDBMS) are an effective tool for most data cleaning operations, and SQL queries provide a natural mechanism for implementing the required filters. Certain data cleaning operations on high frequency 'tick' data are an exception however, and require the use of special software[25].

[25] Dacorogna, M.M., Gencay, R., Muller, U., Olsen, R.B., Pictet, O.V. 2001. *An Introduction to high Frequency Finance*, Academic Press. pp 82-120.

Task: Data Consistency Check

The first and most basic task in the Data Cleaning phase is a detailed check of the domain consistency of the data. The table of summary statistics resulting from the Initial Data Exploration task of the Data Selection phase is a starting point here. The maximum, minimum, range, and variance, of the values for each field in the data set, together with domain knowledge, will help identify the presence of outliers. Each field in the dataset must be individually checked for consistency using filters incorporating appropriate criteria. For example; Call Option prices should be positive, not violate upper or lower bounds[26], and for the same maturity, vary inversely with the strike price. Bid and ask prices should be positive, with the bid price less than the ask price. Implied volatility values should not be unfeasibly large or small. Prices of underlying assets should be consistent with quoted exercise prices. A check for the existence of duplicate records should also be made.

Output: Data Consistency Report

A short report explaining the criteria used to assess data consistency and enumerating the inconsistencies found. The report may include a tabulation of the records containing inconsistencies, provided this is not too large.

Task: De-Duplicate Records, Fill Null Values

Once the data consistency check has been completed the required data cleaning actions can be taken. First and most basic of these are removal of any duplicate records, and the filling of null values.

Output: Dataset free of Duplicate Records and Null Values

A new dataset that is a sub-set of the project dataset with duplicate records filtered out, and empty numeric fields filled.

Task: Delete Inconsistent Prices / Values

Following de-duplication and the filling of null values, remaining inconsistent prices and values are deleted by the application of appropriate filters. RDBMS select queries are an effective tool for this task.

Output: Domain Consistent Dataset

The output of this task is a dataset containing only domain consistent data.

---

[26] Hull, J. 2000. *"Options, Futures and Other Derivative Securities"*, 4th. Ed. Prentice Hall International. p 168-182.

Task: Delete Incomplete Records

The data miner now has a dataset containing only domain consistent data. However, it is possible that a few records remaining in the dataset are incomplete in that fields required as inputs for the data mining phase contain no information. Such records should be deleted to avoid skewing the data mining results.

Output: Dataset of Complete Records

The output of this task is the final result of the Data Cleaning phase. A domain consistent dataset containing only complete records. In this context, a complete record means a record where all variables required as inputs for the data-mining phase contain information.

2.4.3 Data Reduction and Enrichment

The Data Cleaning phase is concerned with removing bad quotes, erroneous values, and unrepresentative outliers, and results in a domain consistent dataset, containing complete records, in the sense defined above. Cleaned datasets are often much smaller than the uncleaned datasets from which they were derived. It is not unusual for up to 90% of the original raw data to be discarded in a data cleaning exercise. Even so, cleaned datasets derived from options or other financial markets, can still be very large. This may lead to problems with file handling or software tools. To deal with this, file sizes are further reduced in the Data Reduction and Enrichment phase by discarding any fields which are not required for the proposed data mining project. However, it may be necessary to add other fields to the project dataset, which are required for the proposed data-mining project, but not present in the original database. An example is interest rates. LIFFE options market data does not include the applicable risk-free interest rate for each trading day. This is required for option pricing, and for calculating implied volatilities. However, it must be extracted from a third-party database and added to the project dataset as an extra field. A RDBMS is an appropriate tool for the Data Reduction and Enrichment phase. Unwanted fields can be filtered out using select queries. New fields can be added by importing the required data as a new table and joining to the table containing the project dataset.

Task: Reduce Dataset Dimensions

Delete all fields not required as inputs for the Data Mining phase.

Output: Dataset Free of Redundant Fields

Project dataset with all its redundant fields deleted.

Task: Enrich with External Data

Add any extra variables (from third-party databases) required as inputs for the Data Mining phase, but not present in the project dataset.

Output: Dataset with Extra Fields from Other Sources

Project dataset free of all redundant fields and complete with any required extra fields containing variables added from third-party databases.

2.4.4 Data Preparation

At this point in the process the data miner is in possession of a clean project data set free of redundancies and enriched with any required additional data obtained from third parties. However, the variables in the dataset may require transformation before they are in the form required for the Data Mining phase. The data miner may wish to input functions of certain variables in the data set rather than the variables themselves. To facilitate this 'derived variables' are created in the dataset, that is new fields containing the desired function values. For example, in option pricing applications it is sometimes desirable to use the ratio (*asset price / exercise price*), termed 'moneyness', as an input rather than the two variables separately. Derived variables that involve combinations of fields in the same record can be straightforwardly created in a RDBMS. Creating more complex derived variables, for example exponential moving averages, involving combinations of fields spanning several records may require coding. Variables must be consistently scaled for the Data Mining phase. Computational knowledge discovery techniques often perform better if the input variables have similar distributions. For this reason it may be desirable to standardise the inputs to minimise the effects of different input dimensions. Finally, the non-parametric nature of many computational knowledge discovery techniques used for data mining means they are prone to over-fitting. To guard against over-fitting, models should be tested on an independent test set, held out from the available data, and used only for testing. The specialised tasks of the Data Preparation phase are thus :

Task: Generate Derived Variables

Derived fields required as inputs for the Data Mining phase are created within the project dataset, using a RDBMS or other appropriate tool.

Output: Dataset Complete with Derived Variables

Project dataset with additional fields containing all required derived variables.

Task: Scale Input Variables

Variables are converted to the scales required for the Data Mining phase by application of an appropriate scale factor.

Output: Dataset with Scaled Variables

Project dataset with all input variables correctly scaled.

Task: Standardise Input Variables

If required, standardise the project dataset to minimise the effect of different input dimensions. The following formula is a suitable transformation, which is widely used for data standardisation. It has the effect of mapping x to the domain [0,1].

$$\text{Standardised}(x_i) = \frac{(x_i - \min(x))}{(\max(x) - \min(x))} \tag{2.1}$$

Equation (2.1) should be applied to all variables in the data set to be standardised including the target (response variable). It may sometimes be useful to normalise inputs to their Z values, which can be accomplished using the following transformation.

$$Z(x_i) = \frac{(x_i - \overline{x})}{\sigma(x)} \tag{2.2}$$

[N.B. If these measures are adopted an inverse transformation should be performed prior to comparing predictions and target values, error measures are only meaningful in terms of the original input dimensions[27].]

Output: Standardised Dataset

A project dataset with all variables standardised.

Task: Partition Data

The project dataset is partitioned into a separate training set and test set by applying an appropriate random sampling algorithm. This ensures that both data sets have distributions representative of the project dataset. [N.B. in the case of ensemble techniques such as bootstrapping, many training sets will be required, created by randomly re-sampling the original training set (with replacement) [28]]

Output: Separate Training Set and Test Set

Two datasets, a training set, and a separate test set independent of that used for training.

---

[27] See e.g. Herrmann, R., and Narr, A. 1997. *"Neural Networks and the Valuation of Derivatives - Some Insights into the implied Pricing Mechanism of German Stock Index Options."* Working paper, University of Karlsruhe, Institute for Decision Theory and Management Science, Department of Finance and Banking. p11.

[28] Efron, B., Tibshirani, R.J. 1993. *An Introduction to the Bootstrap.* Chapman & Hall, New York.

Output: Data Set Description

A description of the data set including the following : The data origins, names and locations (paths) of raw data files, cleaned datasets, reduced and enriched datasets, and final prepared training and test sets. The variables and number of records in the original raw data set. The data cleaning report. Variables which have been added and deleted. Derived variables. Transformation, scaling, and standardisation formulas applied to variables. Method used to partition the data into training and test sets. Variables and number of records included in final training and test sets.

2.4.5 Data Mining

The quality of an option-pricing model depends on its ability to correctly predict option prices. The use of a particular model is only justified if its prediction error is less than that of competing models with the same, or fewer, parameters and inputs. However, in the literature documenting the application of computational knowledge discovery techniques to option pricing, the data mining model search is seldom conducted in a way conducive to statistically valid comparisons of the relative performance of competing models. Because of this, the Data Mining phase of the computational framework is considered here in rather more detail than the other phases.

For neural nets and similar non-parametric non-linear regression techniques, the data mining model search problem can be characterised as follows; Suppose that $\boldsymbol{D} = \{(\boldsymbol{x}_i,\boldsymbol{t}_i);i=1,\ldots,n\}$ is a training dataset containing targets (response variables) $\boldsymbol{t}_i$, and vectors of inputs (explanatory variables) $\boldsymbol{x}_i$, and there exists an unknown true regression

$$\boldsymbol{t}_i = \mu(\boldsymbol{x}_i) + \varepsilon_i \tag{2.3}$$

where $\mu(\boldsymbol{x})$ is the unknown function generating $\boldsymbol{t}$, known as the true *data generating process* (DGP), and ε is a scalar random error term. The set of input vectors $\boldsymbol{X}$ is mapped to $\boldsymbol{t}$ by the *general unrestricted model* (GUM). If a model can account for the findings of other models it is said to *encompass* them. The GUM is said to *nest* the DGP if the latter is a special case of the former. It is assumed that $\boldsymbol{X}$ nests or encompasses some subset of inputs $\boldsymbol{S}$, with significant explanatory power, which is mapped to $\boldsymbol{t}$ by the DGP. The data mining model search problem is to find an estimate

$$\hat{\mu}_\lambda(\boldsymbol{x},\boldsymbol{D}) \tag{2.4}$$

of $\mu(\boldsymbol{x})$, given the training set $\boldsymbol{D}$, and a set of possible model architectures (topologies)

$\mu_\lambda(\boldsymbol{x})$, indexed by λ, in general $\lambda \in \boldsymbol{\Lambda} = (\boldsymbol{S}, A, \boldsymbol{\Omega})$. Here, A is a particular architecture. $\boldsymbol{S}$ is a set of input variables associated with the DGP, and $\boldsymbol{\Omega}$ is a set of weights associated with architecture A and the set of inputs $\boldsymbol{S}$. Thus, the estimate $\hat{\mu}$ depends explicitly on the dataset $\boldsymbol{D}$ used for training, since $\boldsymbol{X} \subset \boldsymbol{D}$ and $\boldsymbol{S} \subset \boldsymbol{X}$, hence $\mathbf{S} \subset \mathbf{D}$. $\hat{\mu}$ also depends explicitly on the selected architecture A, though this may have little effect[29]. The data mining model search therefore consists of a search of the space of input variables and topologies, starting from a carefully selected GUM, with an initial architecture which can accommodate some non-linearity but is not overparameterized. The goal is to identify a terminal model, encompassed by the GUM, in the sense described above, which can generate data that are statistically indistinguishable from the DGP on given criteria. Such a model is said to be *congruent* with the data generating process. If several congruent models exist, a model which encompasses them should be chosen. There should be no other model with fewer inputs or parameters (weights) which fits better. The result is termed the *parsimonious undominated* model[30].

There are two possible strategies for the model search. The first involves a parallel search of both the input space and the space of possible architectures, based on pruning redundant nodes or weights[31]. This approach has the advantage that the data miner is presented with a unique terminal model free of nodes which are redundant on the basis of the pruning criteria. Significance testing is simplified in this case and does not require multiple model adjustments. The disadvantages are that the data miner has no control over which inputs are retained in the model. There is also a lack of control of the selection criteria used at each stage. There is no guarantee that the terminal model represents the end point of a statistically valid search path where all the models are congruent, so that its true significance remains uncertain. Moreover, the approach is computationally costly, as a search for an optimum architecture is carried out for each model generated in the search of the input space.

---

[29] LeBaron, B., and Weigend, A. S. 1998. *"A Bootstrap Evaluation of the Effect of Data Splitting on Financial Time Series"*, IEEE Transactions on Neural Networks 9, p 213-220. The authors present evidence suggesting there is no significant correlation between network performance and architecture or initial weights, on the same training, validation, and test sets, for the kind of data typically found in economics, finance, and business.

[30] Hendry, D. F. 1995. *Dynamic Econometrics*. Oxford: Oxford University Press. Contains extensive discussions of the theory of model reduction.

[31] Automatic sensitivity based pruning is offered as an option in several commercial data mining software packages including SPSS CLEMENTINE.

The second strategy is a sequential search of the input space and the space of potential architectures. This is feasible since the model search described above can be decomposed into a search for the set of significant inputs $\boldsymbol{S}$, and a search for an optimum architecture A (given some $\boldsymbol{S}$). The second strategy has several advantages over the first. It is computationally less costly, since the search for an optimum architecture is only carried out once, for the set of inputs $\boldsymbol{S}$ found in the search of the input space. It amounts to a *coarse* to *fine* development of the model. First, an initial coarse architecture is chosen, and the input space is searched for $\boldsymbol{S}$. Then, the space of feasible architectures is searched for a fine (i.e. in some sense optimal) architecture, starting from the initial coarse architecture. Different methodologies appropriate to each search can be applied. In the search of the input space, the data miner is able to exercise control over the inputs retained in the model, and the variable deletion criteria, and tests, used at each stage. It is possible to ensure that only congruent models are included in search paths, so a better estimate of the true significance of the terminal model is obtained.

In the domain of economic and financial modelling the General-to-Specific (GeTS) procedure has been shown to display superior performance properties, and provide a coherent statistical framework for a model search methodology[32]. However, the GeTS algorithm was developed for use with OLS regression[33]. A model search algorithm based on the GeTS methodology but designed for use with computational knowledge discovery techniques for non-parametric non-linear regression is developed here. The model search algorithm is integrated into the Computational Framework for the input space search task.

Selection of the optimal architecture (topology) is one of the most difficult problems when using neural nets and related techniques for non-linear regression. A sizeable literature on this topic exists, but there is little consensus on the solution[34]. The available approaches can be classified as : a) the heuristic techniques described in the neural net literature, mainly based on pruning. b) Statistical procedures based on hypothesis tests or goodness-of-fit criteria. It is difficult to estimate the true

[32] Hoover, K. D. and Perez, S. J. 1999. *"Data mining reconsidered: Encompassing and the general-to-specific approach to specification search"*. Econometrics Journal 2, p1–25.
[33] Hendry et al (2002).
[34] See e.g. Bishop (1995), Chap.9.

significance of models resulting from approaches of type a). However, this is less of a problem for approaches of type b). To avoid overparameterised models Anders and Korn (1996)[35] suggest hidden layer nodes should only be included when they contribute significantly to the explanation of the response variable. An inclusion criterion based on the estimated prediction error is often used for neural nets and non-parametric models. An architecture selection algorithm based on such a criterion, the Prediction Risk, is presented here. The algorithm is integrated into the Computational Framework for the architecture selection task. The prediction risk contains more information than rival criteria[36], and provides information on the expected error for predictions made by a model. Existing methods of estimating the prediction risk usually rely on cross validation and are computationally costly. A method of estimating the prediction risk that does not require cross validation, is presented in Chapter 3 section 3.4.3.

The data mining model search commences with an initial model choice. Once a terminal model has been selected by the search procedure the final task is to calculate its true significance and performance characteristics. It is important to note that all testing of models must be carried out on independent test data not previously used for model training, or as validation data for early stopping. A description of the tasks and outputs of the data-mining phase follows.

Task: Select General Unrestricted Model and Initial Topology

The data mining model search commences with the selection of a general unrestricted model. Include as inputs, all variables in the data sets produced in the Preparation phase, which could possibly explain the response variable. Next, select an initial topology for the model search. If neural networks are to be used, a network with a single hidden layer containing h nodes should be selected, h is given by;

$$h = \frac{k}{2}, \quad 1 < h < k \tag{2.5}$$

where k is the number of input variables, and h is rounded to the nearest integer. This value is chosen so the network can accommodate some non-linearity in the relationship of inputs to the response variable, while containing as few hidden nodes as possible.

[35] Anders, U., Korn, O., Schmitt, C. 1996. *"Improving the Pricing of Options - A Neural Network Approach"*, ZEW Discussion Paper, pp96-04.

[36] Moody, J.E. 1994. *"Prediction Risk and Architecture Selection for Neural Networks"*, in From Statistics to Neural Networks: Theory and Pattern Recognition Applications, V. Cherkassky, J.H. Friedman and H. Wechsler (eds.), NATO ASI Series F, Springer-Verlag 1994.

The upper bound k on the number of hidden nodes, reflects the empirical observation that neural networks with a non-pyramidal topology have been found to not train as well as networks with a pyramid topology.

Output: Specification for General Unrestricted Model and Initial Topology.

Specification of the input variables, response variable(s), and topology, to be used for the GUM is the starting point for the data mining model search.

Task: Search Input Space (GeTS)

Table 3 shows the General-to-Specific search algorithm, adapted for use with computational knowledge discovery techniques for non-linear non-parametric regression, in a data mining context involving options market data. The algorithm is presented in step-form, in the style of a defining diagram and using a pseudocode-like structured English syntax. The principle underlying the GeTS search algorithm is that inclusion of an insignificant variable in a model is a less serious form of mis-specification than omission of a significant variable. Therefore the GUM is chosen so as to include all potentially significant variables, and the model is progressively simplified by elimination of insignificant variables. The main difficulty in applying the GeTS procedure to NNs and other computational knowledge discovery techniques, is that F and t tests on regression parameters, used for variable deletion tests on linear models, require OLS procedures and therefore cannot be utilised for non-linear techniques[37]. The same is true of the regression diagnostic tests, excepting the Jarque-Bera test for residual normality. There are three well-known tests based on maximum likelihood principles, which can be used for variable deletion with NLLS models. Namely, the likelihood ratio (LR) test, the Wald test, and the Lagrange Multiplier (LM) test. In principle, these can be applied to neural nets provided the assumption of normally distributed residuals holds, and the network weights are uniquely identified (estimable). In practice, the former condition does not usually hold for options market data, and the latter is not met if the network contains redundant hidden layer nodes. Terasvirta et al (1993) and White (1989)[38] describe the procedures required for

[37] Thomas (1997), p 255.

[38] Terasvirta T., Lin C.F., Granger C.W. 1993. *"Power of the Neural Network Linearity Test"*, Journal of Time Series Analysis, 14(2), pp209-220.
White, H. 1989. *"An Additional Hidden Unit Test for Neglected Non-linearity in Multilayer Feedforward Networks"*. Proceedings of the International Joint Conference on Neural Networks, Washington, DC. San Diego: SOS Printing, 11, pp451-455.

Table 3. GeTS Model Search Algorithm for Input Space Search.

INPUT	
Data	**Description**
	a) Training dataset: $\boldsymbol{D} = \{(\boldsymbol{x}_i,\boldsymbol{t}_i);i=1,\ldots,n\}$ with n observations, targets (response variables) $\boldsymbol{t}_i$, and vectors of inputs (explanatory variables) $\boldsymbol{x}_i$. b) Independent test set $\boldsymbol{T} = \{(\boldsymbol{x}_i,\boldsymbol{t}_i);i=1,\ldots,m\}$ to match $\boldsymbol{D}$. (NOTE: Training set $\boldsymbol{D}$ and test set $\boldsymbol{T}$ are outputs of the Data Preparation Phase)
PROCESSING	
Step	**Operations**
1	1.1) Specify General Unrestricted Model (GUM) 1.2) Select initial topology.
2	2.1) Using data set $\boldsymbol{D}$, fit target(s) to the GUM input variables using OLS. Perform F and t-tests on the OLS parameters to suggest the significance of the input variables. 2.2) Perform diagnostic tests, (LM tests for serial correlation, RESET test for functional form, Jarque-Bera test for residual normality, White test for heteroskedasticity). 2.3) IF diagnostic test results suggest non-linear model appropriate GOTO **Step 3** ELSE apply GeTS procedure for OLS ENDIF
3	3.1) Train network using dataset $\boldsymbol{D}$, GUM specification, and initial topology. 3.2) Using dataset $\boldsymbol{T}$, perform F and t-tests on the actual and fitted values of the targets. Apply Jarque-Bera test for residual normality, if hypothesis of normality is rejected, do not repeat this test for subsequent steps. 3.3) IF the F and t statistics suggest there is no significant difference between the actual and fitted values of the target, designate the GUM as the Current Model (CM) GOTO **Step 4** ELSE GOTO **Step 1** ENDIF
4	4.1) Specify a Restricted Model (RM) by deleting input variable(s) from the CM. Using dataset $\boldsymbol{D}$, train a network using the RM specification and initial topology. 4.2) Using dataset $\boldsymbol{T}$, perform F tests and paired t-tests, on the residuals from the CM and the RM. Apply Jarque-Bera test for residual normality, if hypothesis of normality is rejected, do not repeat this test for subsequent steps. 4.3) IF hypothesis of no difference between the residuals is accepted in the F and t tests, designate the RM as the next CM REPEAT **Step 4** ELSE discard RM ENDIF 4.4) REPEAT **Step 4** UNTIL restrictions / search paths exhausted. GOTO **Step 5**
5	5.1) IF number of terminal models generated by **Step 4** >1 Specify an encompassing model (EM). Using dataset $\boldsymbol{D}$, train a network using the EM specification, and initial topology ELSE GOTO **Step 6** ENDIF 5.2) Using dataset $\boldsymbol{T}$, perform F and t-tests on the actual and fitted values of the target. Apply Jarque-Bera test for residual normality, if hypothesis of normality is rejected, do not repeat this test for subsequent steps. 5.3) IF the F and t statistics suggest there is no significant difference between the actual and fitted values of the target, designate the EM as the CM GOTO **Step 4** ELSE discard EM. ENDIF 5.4) REPEAT **Step 5** UNTIL a PARSIMONIOUS UNDOMINATED EM is obtained.
6	6.1) GOTO Architecture Selection Algorithm.
OUTPUT	
Data	**Description**
	c) Parsimonious undominated model, $\hat{\mu}\ (\boldsymbol{x},\boldsymbol{S})$, congruent with the true Data Generating Process. d) Identity of those input variables, (comprising subset $\boldsymbol{S}$ of the training dataset $\boldsymbol{D}$), which have significant explanatory power for the target(s) $\boldsymbol{t}_i$.

constructing the test statistics, they are complex and cannot be used with standard neural net software.

A limited range of tests is therefore available for use with NNs or related techniques. The Jarque-Bera test for residual normality can be used to indicate model mis-specification. Non-normality of the residuals means the NN is not an efficient estimator, though it remains consistent and asymptotically unbiased. Plots of residuals or squared residuals against each input variable, must be relied on to indicate the existence of heteroskedastic error terms. F tests and paired t tests on the predicted values and the residuals can be used for model comparison. If a significant variable is eliminated from a model, there will be a significant change in the residuals at each observation. If an insignificant variable is eliminated, then there will be no significant change at each observation. Thus, the modified GeTS algorithm shown in Table 3, employs F tests and paired t-tests of the residuals for variable deletion tests[39].

Output:

Model(s) that are congruent parsimonious undominated encompassing models of the DGP, nested within the GUM.

Task: Search for Optimal Architecture

Techniques for selecting the optimal architecture for a NN can take the form of *pruning algorithms* or *growing algorithms.* With pruning algorithms, a large network is first trained, and redundant hidden layer nodes (and usually input variables) are eliminated by the application of a selection criterion. The criterion used can be either statistical, or a version of sensitivity based pruning. It is difficult to determine the true statistical significance of the terminal model when sensitivity based pruning is used. This is because sensitivity based pruning is an automatic procedure which works by eliminating weights, hidden layer nodes, and inputs, if the partial derivatives of the network outputs with respect to them are below some arbitrarily selected small threshold value. Moreover, pruning algorithms require the estimation of models with redundant weights, consequently some weights are unidentified, so the assumptions underlying the usual statistical tests are violated[40].

[39] See Appendix F for details of the statistical tests discussed here.

[40] Anders, U. 1996. *"Statistical Model Building for Neural Networks"*, Proceedings of the 6th International AFIR Colloquium, Nürnberg, Germany, October 1-3, 1996. p969.

To avoid these problems, a growing algorithm is used to select an optimal architecture with minimal or no redundant hidden layer nodes. The selection criterion used at each step is the prediction risk (PR) criterion[41]. The PR is chosen because the goal of network training is to find a network that gives good predictions on new (unseen) data. That is, a network with low variance which generalises well. The PR is the expected mean squared error (*E[MSE]*) of an estimator in predicting new observations, and is thus a measure of generalisation ability. PR is a summary statistic closely related to prediction intervals. Prediction intervals can be used as an alternative instrument for network architecture selection[42], however PR is more straightforward to use. Prediction intervals, and PR, are discussed in detail in Chapter 3, methods of estimating them are discussed, and a new approach suitable for use with large databases of option market data is presented.

Table 4. shows the Architecture Selection Algorithm. This is a heuristic algorithm described in terms of NNs, but it can be used with most computational knowledge discovery techniques for non-linear non-parametric regression.

The algorithm shown in Table 4 has the advantage that a sequence of 'nested' models is generated, permitting estimation of the true significance of the final results of the full model search process. Prediction Risk is normally calculated using independent test set data, or individual observations randomly omitted from the training data (cross-validation). It does not decrease monotonically as the number of hidden layer nodes is increased, because of the existence of local minima, as discussed by Moody (1994). For this reason the algorithm is *not* terminated as soon as PR increases. Instead, sufficient iterations of Step 2 are performed to create a set of architectures that might reasonably be expected to span the optimal architecture. The model with the smallest PR is then identified. It is selected as the 'optimal' architecture, unless there is another model with fewer hidden layer nodes and a PR which is little different,.

Output:

Congruent parsimonious undominated encompassing model(s) of the DGP, with optimised architecture, having minimal or no redundant weights.

---

[41] Moody, J. 1994. *"Prediction Risk and Architecture Selection for Neural Nets"*, in *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, V. Cherkassky, J.H. Friedman and H. Wechsler (eds.), NATO ASI Series F, Springer-Verlag.

[42] Hwang, J.T.G. and Ding, A.A. 1997. *"Prediction Intervals for Artificial Neural Networks"*, Journal of the American Statistical Association, Vol. 92 No. 438, p754.

Table 4. Architecture Selection Algorithm

INPUT	
Data	**Description**
	e) Training dataset: $\boldsymbol{D} = \{(\boldsymbol{x}_i,t_i);i=1,\ldots,n\}$ with n observations, targets (response variables) t_i, and vectors of inputs (explanatory variables) $\boldsymbol{x}_i$. f) Independent test set $\boldsymbol{T} = \{(\boldsymbol{x}_i,t_i);i=1,\ldots,m\}$ to match $\boldsymbol{D}$. g) Parsimonious undominated model, $\hat{\mu}\ (\boldsymbol{x},\boldsymbol{S})$, from Input Space Search Algorithm. h) Maximum numbers of hidden layer nodes to be considered. (NOTE: Training set $\boldsymbol{D}$ and test set $\boldsymbol{T}$ are outputs of the Data Preparation Phase)
PROCESSING	
Step	**Operations**
1	1.1) Designate the parsimonious undominated model, $\hat{\mu}\ (\boldsymbol{x},\boldsymbol{S})$, as the Current Model (CM). 1.2) Using dataset $\boldsymbol{T}$, Estimate the PR for the CM.
2	2.1) Add ONE hidden layer node to the network architecture of the CM. Train a new model with this revised architecture, using dataset $\boldsymbol{D}$. (Freeze the random starting weights, or use estimated weights from the CM as starting weights.) 2.2) Using dataset $\boldsymbol{T}$, estimate the PR for the new model. 2.3) REPEAT **Step 2** UNTIL number of hidden layer nodes = maximum.
3	3.1) Select model with the *smallest* PR from the set of models created in **Step 2**. Designate as CM. 3.2) IF in the set of models there is a model with *fewer* hidden layer nodes than the CM, AND its PR is not significantly different from the CM. Select this model as the Optimal Model. ELSE select CM as the model with optimal architecture ENDIF
OUTPUT	
Data	**Description**
	i) Parsimonious undominated model with optimised architecture, $\hat{\mu}_{\lambda}(\boldsymbol{x},\boldsymbol{D})$. (NOTE: $\lambda \in \boldsymbol{\Lambda} = (\boldsymbol{S},A,\boldsymbol{\Omega})$, where A is a particular architecture, (in this case the optimised architecture). $\boldsymbol{S}$ is the set of input variables associated with the true Data Generating Process, and $\boldsymbol{\Omega}$ is the set of weights associated with architecture A and the set of inputs $\boldsymbol{S}$)

Task: Evaluate Terminal Model(s)

Evaluate the performance and estimate the true significance of the final model(s) resulting from the complete model search. Take account of the numbers of hypothesis tests performed and whether models are nested within one another. Assuming the models are all nested within one another and j steps are required to reach the final model, then the true significance of the j^{th} hypothesis test in the sequence is given by

$$\alpha^* = 1-(1-\alpha)^j \tag{2.6}$$

where α is the nominal significance level[43]. It is usual to test for a given level of significance, normally $\alpha^* = 0.05$. In order to ensure that all tests in a sequence have a true significance of $\alpha^* = 0.05$, the nominal significance must be adjusted for each j by

[43] Maddala, G. S. 1988. *Introduction to Econometrics*, New York, Macmillian, p425.

solving the following equation for α.

$$0.05 = 1-(1-\alpha)^j \tag{2.7}$$

Table 5 shows true and nominal significance for a sequence of 10 consecutive hypothesis tests, obtained by solving (2.6) and (2.7), assuming nominal and true significance of 0.05 respectively.

Table 5. Nested Models: True and Nominal Statistical Significance

j	True α^* for $\alpha = 0.05$	Nominal α for $\alpha^* = 0.05$
1	0.050	0.0500
2	0.098	0.0253
3	0.143	0.0170
4	0.185	0.0127
5	0.226	0.0102
6	0.265	0.0085
7	0.302	0.0073
8	0.337	0.0064
9	0.370	0.0057
10	0.401	0.0051

Output: Model Performance Statistics and True Significance Levels

Performance statistics, and estimates of the true statistical significance, for final model(s) resulting from the data-mining model search.

2.4.6 Reporting and Deployment

The final phase in the Specialised Process Model for Options Market Applications is Reporting and Deployment. The final report is a record of the findings of the entire KDD and data mining exercise. It is essential for management decisions regarding operational deployment of the models developed in the Data Mining phase. It is also a useful guide for similar exercises in the future.

User manuals will be required for any models which are to be deployed throughout the enterprise. Also, the operational performance of models needs to be monitored, so that models can be replaced or updated when necessary. This is important in an options market context, as the performance of models may decay quickly due to the non-stationary nature of market data. It is best if a systematic monitoring and maintenance plan is in place at the time of deployment.

The Reporting and Deployment Phase has the following tasks :

Task: Prepare Statistics and Graphics

Collate and tabulate all statistics produced in the different Phases of the process that are to be included in the final report. Produce all necessary associated graphical presentations.

Output: Tables and Figures for Final Report

All tables of statistics, figures, and graphs, to be included in the final report document.

Task: Recommend Deployment / Non-use

Prepare advice to management for inclusion in the final report, in favour or against deployment of the models developed, based on performance test results and statistics.

An estimate of model risk[44], the risk to the business of loss due to weakness of the model should also be included.

Output: Deployment Advice

Estimates of model risk, and advice to management concerning deployment of the model(s) for inclusion in the final report.

Task: Produce Final Report

Prepare the final report document, include the tables and figures, and the deployment advice, from the previous steps. The document should take the form of a short formal report, and be based on a standard template.

Outputs:

Final Report Document

Final report document, including executive summary.

Users Manual

Contingent on a recommendation to deploy the model(s), prepare a users manual. The users manual should be a short manual describing the model, specifying its performance characteristics and limitations, and precisely stating the circumstances when it should or should not be used.

[44] See e.g. Rebonato, R. 2003. *"Theory and Practice of Model Risk Management"*, Quantitative Research Centre (QUARC) of the Royal Bank of Scotland, Oxford Financial Research Centre – Oxford University.

Task: Plan Monitoring and Maintenance

Subject to a decision to deploy the model(s), prepare a Monitoring and Maintenance Plan. The plan should be developed following consultation with the intended users. Users should be required to keep a log of model use, and make periodic performance reports to the development team. Suitable forms should be designed for the purpose. The maintenance plan should take account of the expected life cycle of the model, including its eventual retirement and replacement.

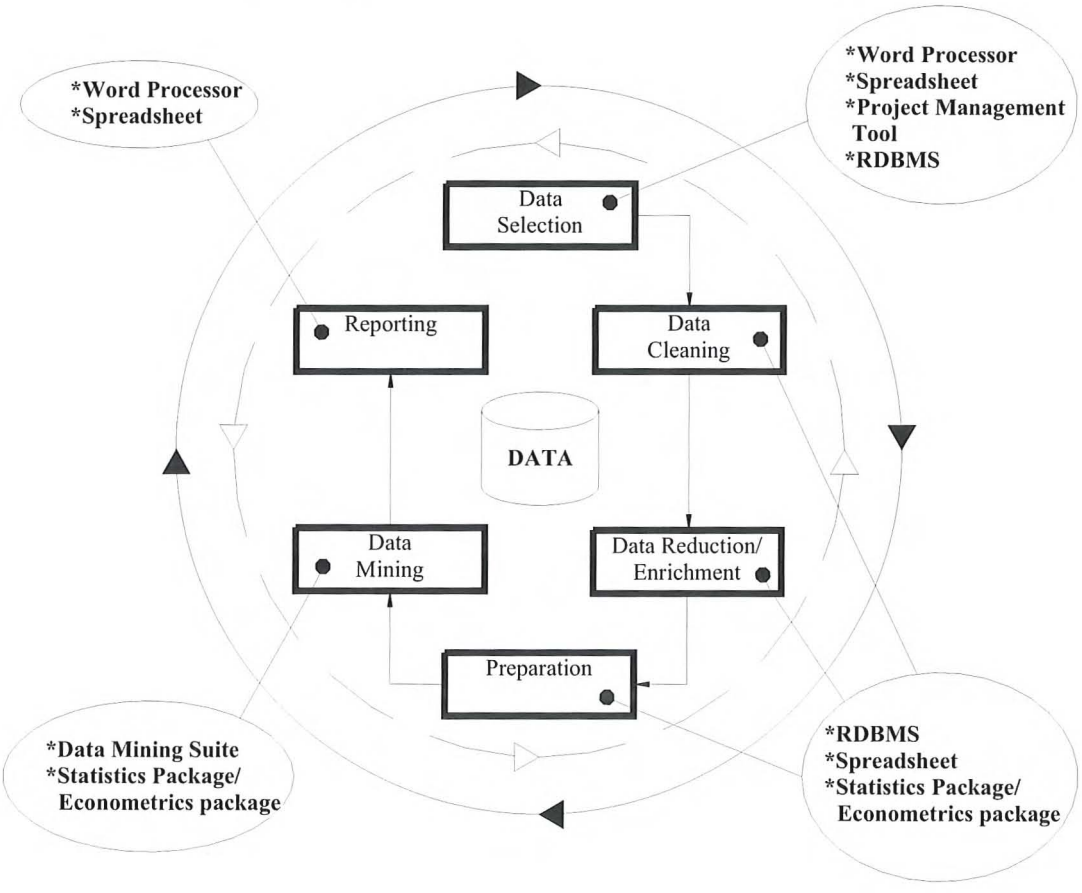
Output: Monitoring and Maintenance Plan.

A Monitoring and Maintenance Plan for the full model life cycle.

2.5 Multiple Iterations and Process Instances

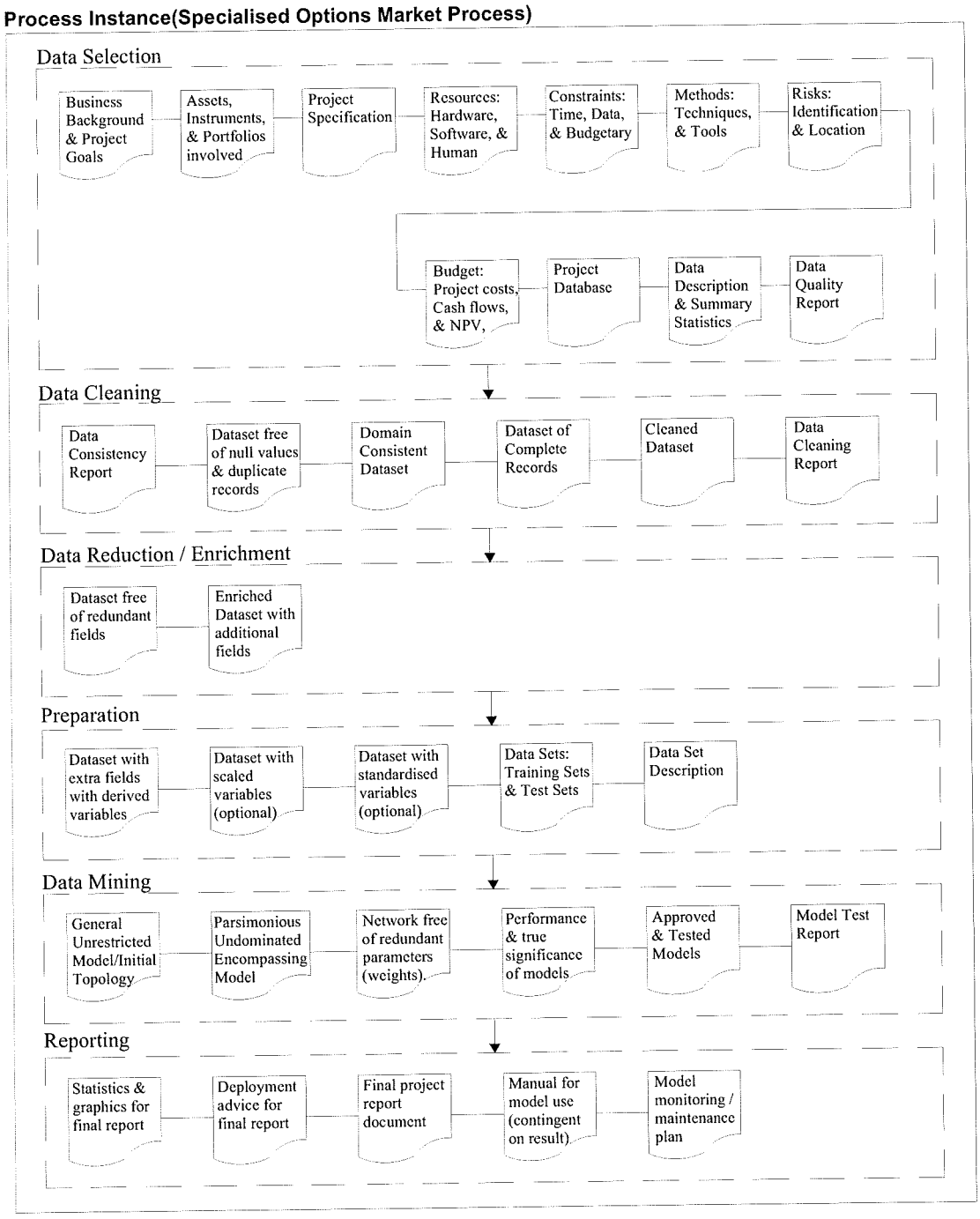
Fig.6 illustrates an idealised single iteration through the Specialised Process Model for Options Market Applications. The sequence of phases and tasks is designed to minimise repetition. However, multiple iterations through all or part of the process may still be required to attain a particular KDD or data mining goal. Some steps in the process may need to be repeated more than once. For example, to introduce new data in the Data Reduction / Enrichment phase, it may be necessary to cycle through the Data Cleaning phase again. The sequence of phases and tasks illustrated in Fig.6 should not therefore be thought of as fixed. Rather, the Process Model overall is cyclical not linear, with many possible pathways through it. Fig.7 is a diagrammatic representation of multiple iterations through the process. The outer circle with solid arrows represents the cyclical nature of the process, and the inner circle represents a feedback loop. The actual software implementation of the full Specialised Options Market Process may require the use of multiple software tools for each phase, in addition to specialised software for the data mining phase, and these are shown.

Fig.7 Specialised Options Market Process: Phases and Tools



A Process Instance is a record of a single iteration through the Process Model. It is a record of actions, decisions, and results, and consists of all of the outputs from all of the tasks completed for each phase. Fig.8 overleaf illustrates the content of a process instance for a typical single iteration through the full Specialised Process Model for Options Market Applications.

Fig.8 Specialised Options Market Process: A Process Instance



2.6 Conclusions

A framework for applying computational knowledge discovery techniques to options market data in a manner designed to avoid attributing significance to spurious relationships was developed and presented in Chapter 2. Procedures and methods were described to estimate true *global* statistical confidence in models, and diagnose problems that may affect model assumptions, for example tests of functional form, and the normality of residuals. The testing procedures described in this chapter, reveal little about how a model will perform on new *unseen* data, however. Also, the nature of financial data, and the input dimensionality and function complexity of models used in options markets, make it desirable to obtain *local* estimates of confidence. These central questions of how to estimate the confidence, which can be placed in models and predictions are addressed in Chapter 3.

CHAPTER 3

Confidence in Models and Predictions

3.0 Introduction

In this chapter, a theoretically well founded and robust method for determining prediction intervals, and prediction risk, useable with computational knowledge discovery techniques for non-parametric non-linear regression is presented. The exposition is based on arguably the most successful type of computational knowledge discovery technique for options market applications. Namely, the form of neural network termed a multi layer perceptron (MLP). However, it is shown here that the method is equally applicable to a broad class of related techniques. The method was subjected to extensive empirical testing using a standard synthetic data set, and compared with a method applicable only to the MLP. Following this, to assess its performance in more realistic settings, the method was applied to synthetic option prices, and then to observed market prices of options.

3.1 Motivation

Neural nets are a flexible computational knowledge discovery technique widely used to model high-dimensional real-valued non-linear data. Hornik et al (1989) have shown that an MLP with a single hidden layer can approximate arbitrarily closely, virtually any linear or non-linear continuous function. They also demonstrate[45] that MLPs are twice differentiable so first and second order partial derivatives of the network with respect to its inputs are obtainable. These characteristics make MLPs suitable for applications in option pricing, hedging, and the recovery of probability distributions from option market data. This is supported by Galindo[46] who presents evidence suggesting that MLPs out-perform other computational knowledge discovery techniques for option pricing. A number of researchers have applied neural nets to option pricing and report favourable results, compared with modern parametric option pricing models[47]. Surprisingly, there has been little reported for the confidence factors associated with the modelling of option prices using neural nets. Even more surprising is the absence from the literature of standard statistical hypothesis testing[48], so comparisons between models

[45] Hornik, K., Stinchcombe, M., White, H. 1990. "*Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks*", Neural Networks, Vol. 3, pp 551-560.
[46] Galindo (1999).
[47] See Literature Review Chapter 4, section 4.1.2
[48] Sutcliffe, C. Private Communication. January 2003. I am grateful to Professor Sutcliffe for confirming this observation.

are made on measures such as mean-squared-error and R^2, which give an incomplete picture of performance.

Typical NN models contain large numbers of weights and bias terms (parameters). The number of these often greatly exceeds the number of input variables. Because a NN is a linear combination of large numbers of basis functions, these parameters are uninterpretable. They are frequently also unidentified. Hence, neural nets are often regarded as "black box" models[49]. This may explain the tendency among users to ignore the uncertainty inherent in predictions produced by the networks. However, like other regression methods, neural net prediction accuracy varies with data density and noise. Also, in options market applications particular models can often be statistically acceptable for a set of data, but if inputs are partitioned, for example, by moneyness and maturity, then some partitions fail the statistical tests. Consideration of error estimates on a point by point basis is therefore essential if a fuller picture of performance is to be obtained. Moreover, prediction intervals, and the closely related prediction risk, are useful statistical criteria for application to the model selection problem, as discussed in Chapter 2 section 2.4.5.

[49] The structure and functioning of neural nets is explained in Appendix A.

3.2 Theory

This section is mainly pedagogical. To define terms and develop notation the theory relating to confidence and prediction intervals applied to regression is first reviewed. A discussion of how this theory applies to neural nets then follows.

3.2.1 Confidence and Prediction Intervals for Regression

Regression is the name given to the family of statistical techniques used to model the relationship between a *response* (or *dependent*) *variable* y, and a vector $\boldsymbol{x}$ of *explanatory* (or *independent*) *variables*, the *regressors*. In the neural net literature the terminology *targets* for response variables, and *inputs* for the regressors, with the vector $\boldsymbol{x}$ termed the *input vector* is used. The following discussion adopts this usage. In regression, it is assumed there is a relationship between the target y and the input vector $\boldsymbol{x}$. Equation (3.1) shows a possible form this may take.

$$y = \mu_y(\boldsymbol{x}) + \varepsilon \tag{3.1}$$

The relationship has both stochastic and deterministic components. Here $\varepsilon \sim \mathrm{N}(0,\sigma^2)$ is a normally distributed random error. The stochastic component consists of the resulting random fluctuation of y about its mean $\mu_y(\boldsymbol{x})$. The deterministic component is the function relating $\mu_y(\boldsymbol{x})$ and $\boldsymbol{x}$. Suppose that the true but unknown function relating $\mu_y(\boldsymbol{x})$ and $\boldsymbol{x}$ is given by

$$\mu_y(\boldsymbol{x}) = f(\boldsymbol{x};\boldsymbol{\beta}) \tag{3.2}$$

where $\boldsymbol{\beta}$ is a set of parameters. Regression attempts to model this relationship by estimating the parameter values from the data set. To achieve this, the values of $\boldsymbol{\beta}$ are adjusted under the assumption that f is the true function, giving

$$\hat{\mu}_y(\boldsymbol{x};\hat{\boldsymbol{\beta}}) = f(\boldsymbol{x};\hat{\boldsymbol{\beta}}) \tag{3.3}$$

where a hat denotes an estimated value. The right hand side of equation (3.3) is termed a *regression function*. If $\hat{\mu}_y(\boldsymbol{x};\hat{\boldsymbol{\beta}})$ is estimated from a finite sample $\boldsymbol{S}$, $\{(\boldsymbol{x}_1,y_1), (\boldsymbol{x}_2,y_2), \ldots, (\boldsymbol{x}_n,y_n) \in \boldsymbol{S}\}$, sampling variation in $\boldsymbol{S}$ will result in variation in $\hat{\boldsymbol{\beta}}$ and hence variation in $\hat{\mu}_y(\boldsymbol{x};\hat{\boldsymbol{\beta}})$. It follows $\hat{\mu}_y(\boldsymbol{x}_0;\hat{\boldsymbol{\beta}})$ has a sampling distribution about $\mu_y(\boldsymbol{x}_0)$, where $\boldsymbol{x}_0$ is a particular value of $\boldsymbol{x}$. A 95% confidence interval for $\mu_y(\boldsymbol{x}_0)$ is an interval $[\lambda_L(\boldsymbol{S}, \boldsymbol{x}_0), \lambda_U(\boldsymbol{S}, \boldsymbol{x}_0)]$ about $\hat{\mu}_y(\boldsymbol{x}_0;\hat{\boldsymbol{\beta}})$ such that $\mu_y(\boldsymbol{x}_0)$ is within the interval in 95% of cases. A 95% prediction interval is an interval $[\psi_L(\boldsymbol{S}, \boldsymbol{x}_0), \psi_U(\boldsymbol{S}, \boldsymbol{x}_0)]$ about $\hat{\mu}_y(\boldsymbol{x}_0;\hat{\boldsymbol{\beta}})$ such that

the unknown value y_0 associated with $\boldsymbol{x}_0$ is within the interval in 95% of cases. As an example consider the univariate ordinary least squares (OLS) regression of y on x for a sample $\boldsymbol{S}$, $\{(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n) \in \boldsymbol{S}\}$. The 95% confidence interval for $\mu_y(\boldsymbol{x}_0)$ is given by

$$\hat{\mu}_y(x_0;\hat{\beta}) \pm t_{.025(n-2)}\left(s_y\sqrt{\frac{1}{n}+\frac{(x_0-\bar{x})^2}{\sum_{i=1}^{n}(x_i^2)}}\right) \tag{3.4}$$

and the 95% prediction interval by

$$\hat{\mu}_y(x_0;\hat{\beta}) \pm t_{.025(n-2)}\left(s_y\sqrt{\frac{1}{n}+\frac{(x_0-\bar{x})^2}{\sum_{i=1}^{n}(x_i^2)}+1}\right) \tag{3.5}$$

where s_y is the standard deviation of the y values and $\bar{x}$ is the mean of the x values[50]. In equations (3.4) and (3.5), s_y is given by

$$s_y = \sqrt{\frac{\sum e_i^2}{n-2}} \tag{3.6}$$

where $e = (\hat{\mu}_y(x_i;\hat{\beta}) - y_i)$ are the residuals. This follows from the classical assumptions for OLS regression under which $\mathrm{Var}(y_i) = \mathrm{Var}(\varepsilon_i) = \sigma^2$. An unbiased estimate of σ and hence of the standard deviation of the y_i is given by equation (3.6). Equations (3.4) and (3.5) can be generalised to the multivariate case for OLS regression to obtain a (1-α)100% confidence interval for $\mu_y(\boldsymbol{x}_0)$. In matrix notation this is

$$\hat{\mu}_y(\boldsymbol{x}_0;\hat{\boldsymbol{\beta}}) \pm t_{(\alpha/2)(n-k-1)}\left(\sqrt{\boldsymbol{x}_0^T(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_0 s_y^2}\right) \tag{3.7}$$

and a (1-α)100% prediction interval is

$$\hat{\mu}_y(\boldsymbol{x}_0;\hat{\boldsymbol{\beta}}) \pm t_{(\alpha/2)(n-k-1)}\left(\sqrt{(1+\boldsymbol{x}_0^T(\boldsymbol{X}^T\boldsymbol{X})^{-1})\boldsymbol{x}_0 s_y^2}\right) \tag{3.8}$$

In equations (3.7) and (3.8) $\boldsymbol{x}_0$ is a $k \times 1$ column vector of inputs and $\boldsymbol{X}$ is a $n \times k$ matrix, containing first a column of ones, and then the $k-1$ values of each of the n row vectors $\boldsymbol{x}_i^T$. The scalar s_y^2 is the mean squared residual (MSR). It is an unbiased estimate of σ^2 and hence of the variance of the y_i. For equations (3.7) and (3.8), s_y^2 is given by

$$MSR = s_y^2 = \frac{\boldsymbol{Y}^T\boldsymbol{Y} - \hat{\boldsymbol{\beta}}^T\boldsymbol{X}^T\boldsymbol{Y}}{n-k-1} \tag{3.9}$$

[50] Thomas (1997), pp150-153.

In equation (3.9), $\boldsymbol{X}$ is as previously defined, $\boldsymbol{Y}$ is a $n \times 1$ vector of target values, and $\hat{\boldsymbol{\beta}}$ is a $k \times 1$ vector of estimated parameters given by $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$. If the x values in equations (3.4) to (3.9) are continuously valued over the interval $[x_1, x_2]$, $x_2 > x_1$ a continuous *confidence band* and *prediction band* are obtained. From equation (3.2) $f(\boldsymbol{x};\boldsymbol{\beta})$ is the true but unknown function relating $\mu_y(\boldsymbol{x})$ and $\boldsymbol{x}$ (the *true regression*). Equation (3.3) $\hat{\mu}_y(\boldsymbol{x};\hat{\boldsymbol{\beta}}) = f(\boldsymbol{x};\hat{\boldsymbol{\beta}})$ is an estimate of this regression. It follows that the confidence intervals (3.4) and (3.7) are for the true regression functions. The prediction intervals (3.5) and (3.8) are for predicted values associated with a new unseen input. The relationship between confidence intervals and prediction intervals can be understood by considering the following equation.

$$[y - f(\boldsymbol{x};\hat{\boldsymbol{\beta}})] = [f(\boldsymbol{x};\boldsymbol{\beta}) - f(\boldsymbol{x};\hat{\boldsymbol{\beta}})] + \varepsilon(\boldsymbol{x}) \tag{3.10}$$

Prediction intervals are concerned with the left hand side of this equation, the difference between the target value and its predicted value from the estimated regression function. The left hand term decomposes into the two right hand terms. Confidence intervals are concerned with the first of these, the difference between the true and estimated regression functions. This difference is determined by the parameter difference $(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})$. The remaining term on the right hand side is the (sample dependent) random noise term. Neither the noise or the true parameters $\boldsymbol{\beta}$ can be directly observed. However, the variance of the noise can be estimated as $\widehat{Var}(\varepsilon) = s^2$, and confidence intervals constructed for the true $\boldsymbol{\beta}$.

3.2.2 Estimating Confidence and Prediction Intervals for NNs

The theory presented in the previous section can be applied where the right hand side of equation (3.3), the regression function, is a neural net. Existing methods are discussed in this section. The discussion is confined to the case of a MLP with one hidden layer. In this case, from equation (3.3),

$$\hat{\mu}_y(\boldsymbol{x};\hat{\boldsymbol{\Omega}}) = \Theta\left(\sum_{h=1}^{H} w_{lh}\Phi_h\left(\sum_{k=1}^{K} w_{hk}x_k + \omega_h\right) + \omega_l\right) \tag{3.11}$$

In equation (3.11), the MLP consists of one layer of K input nodes $x_1, \ldots, x_K$, a layer of l output nodes, and H hidden layer nodes[51]. The functions Θ and Φ are termed activation functions. For the hidden layer, Φ is usually a sigmoid function such as the

[51] Bishop (1995), pp 117-119.

logistic function or the hyperbolic tangent function. For a continuously valued target variable, the output functions Θ are usually linear and may be the identity. The w are referred to as the weights and the ω are constant intercept terms known as biases. The set of estimated weights and biases is denoted by $\{\hat{\Omega}, (w_1,\ldots,w_{KH+Hl}, \omega_1,\ldots,\omega_{H+l},) \in \hat{\Omega}\}$. Unlike conventional NLLS regression, in neural nets the non-linearity involved is not parametrically defined *a priori*, and can assume any form. Consequently, the error surface can have many local minima, and a closed form solution for the global minimum is not generally possible. The network is fitted ('trained' in the neural net literature) by searching a weight space to select $\hat{\Omega}$ to minimise a cost function. This is done by employing a search algorithm such as gradient descent, conjugate gradient, or quasi-Newton[52]. These methods require initialisation of the vector Ω with a set of small random values. The vector $\hat{\Omega}$ that minimises the cost function is then iteratively estimated. Because the vector $\hat{\Omega}$ is typically large, search methods involving computation of the Hessian matrix of second order partial derivatives of the cost function with respect to each of the elements of $\hat{\Omega}$ are computationally intensive. In common with other non-parametric methods, neural nets can over-fit data. To prevent over-fitting, training is terminated when the error function is minimised on a validation data set (*early stopping*). Alternatively, some form of regularisation (*weighting*) can be used. See Appendix A for further explanation of the workings of MLPs.

When the cost function used for (3.11) is the sum of squared errors, the activation functions are not all linear (the usual case), and early stopping is used to prevent over fitting, the optimisation is effectively a NLLS regression. The theory for estimating standard errors for non-linear regression is then directly applicable. Hwang and Ding (1997)[53] show this theory can be extended to MLPs to obtain asymptotically correct standard errors. The estimation of standard errors for MLPs using the Delta Method[54] and Sandwich Method[55] for non-linear regression is explained in Appendix B. Alternative bootstrap and Bayesian approaches are explained in Appendix C.

---

[52] Bishop (1995), pp 253-294.
[53] Hwang et al (1997).
[54] Seber, G.A.F., and Wild, C.J. 1989. Nonlinear regression, John Wiley & Sons, New York.
[55] Huber, P. J. 1967. *"The behavior of maximum likelihood estimation under nonstandard conditions"*, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, LeCam, L. M. and Neyman, J. editors. University of California Press, pp 221-233.
White, H. 1982. *"Maximum likelihood estimation of misspecified models"*. Econometrica, 50, pp 1-25.

3.3 Literature Review: Confidence and Prediction Intervals for NNs

In this section, the literature on the performance of existing methods of estimating standard errors for NNs is reviewed. Recent research, aimed at obtaining improved estimates of standard errors, confidence, and prediction intervals for neural nets, is also discussed.

3.3.1 Existing Methods of Estimation

Tibshirani (1996)[56] has performed tests of seven different standard error estimates for MLPs with single hidden layers and a linear output layer. These were obtained using three variants of the delta method, two variants of the sandwich estimator, and the bootstrap pairs and bootstrap residuals methods. The delta method variants used are the standard, a method using the inverse Hessian matrix, a method using an approximation to the Hessian matrix omitting second order terms, and the delta method with a regularisation term. The sandwich method variants are the standard sandwich method and a variant using an approximate Hessian matrix.

A small data set was used for these tests, consisting of 111 observations on air pollution. For the bootstrap methods, B = 20 bootstrap replicates were used. In these tests, it was found that the bootstrap methods provided the most accurate estimates of standard errors. The delta methods and sandwich estimators missed the substantial variability due to random starting weights. Tibshirani suggests these latter estimators may perform better where there is less sensitivity to the choice of starting weights (initialisation parameters), for example with larger data sets where gradient descent is used.

This suggestion is consistent with findings by LeBaron and Weigend (1994)[57] who used a training set of 3200 observations on market trading volume at the New York Stock Exchange. This data was relatively noisy, with predictions explaining approximately 0.5 of the variance. They created 2523 bootstrap replicates for both an MLP and a linear model. The error measure used was $(1\text{-}R^2)$, and 2523 bootstrap replicates were generated on the test set of 1500 observations to obtain out-of-sample distributions for

[56] Tibshirani, R., 1996. *"A comparison of some error estimates for neural network models"*. Neural Computation, 8, pp 152-163.
[57] LeBaron, A. and Weigend, A. 1994. *"Evaluating Neural Network Predictors by Bootstrapping"*, In: Proceedings of the International Conference on Neural Information Processing (ICONIP'94), Seoul.

this error. For the MLP, 697 networks were also trained on a single sample and initialisation parameters were randomly drawn for each one. It was found that the randomness due to the splitting of the data generated more variability than the randomness due to network initialisation. Indeed, no significant correlation was found between the choice of initialisation parameters or network topology, and performance. Moreover, the error distributions obtained from the bootstrap procedure on the test set are almost identical for both the linear and MLP models. This suggests that for the data used, the MLP was unable to extract non-linear features that generalised out-of-sample from the (noisy) training set.

In tests using synthetic data with an input-dependant noise variance, Bishop and Qazaz (1995)[58] demonstrate the Bayesian approach can give an improved estimate of noise variance compared with maximum likelihood based approaches. However, Ungar et al (1995) question whether the improved performance of Bayesian approaches justifies the extra computational cost involved[59].

Heskes (1997)[60] proposes a method based on bootstrap pairs for obtaining prediction intervals for a pair $\{(\boldsymbol{x}_0, y_0) \notin \boldsymbol{S}_b^{Boot}, b = 1 \text{ to } B\}$ using the relationship in equation (3.10) , and equations (C.1) and (C.2) given in Appendix C. To achieve this a separate neural net $\chi^2(\boldsymbol{X})$ is trained to model the noise variance $Var(\varepsilon)$. The targets for this network are residuals satisfying

$$r^2(\boldsymbol{X}_v) = Max\left(e^2_{Boot}(\boldsymbol{X}_v) - \widehat{SE}^2_{Boot}(f(\boldsymbol{X}_v;\boldsymbol{\Omega})), 0\right) \tag{3.12}$$

obtained from the validation sets used for training the B networks used in the bootstrap ensemble (C.2). Alternatively, these may be obtained by applying (C.2) to an independent test set. In equation (3.12), $e^2_{Boot}(\boldsymbol{X}_v) = \left(y - \hat{\mu}_{y,Boot}(\boldsymbol{X}_v)\right)^2$, the residuals from (C.2) when applied to the validation sets. The cost function used for training the auxiliary network is the negative log likelihood function, hence the use of the $Max(\bullet,0)$ function in (3.12). The resulting bootstrap prediction interval is

[58] Bishop, C.M., Qazaz, C.S. 1995. *"Bayesian Inference of Noise Levels in Regression"*, Technical Report, Neural Computing Research Group, Aston University.
[59] Ungar, L.H., De Veaux, R.D., Rosengarten, E. 1995. *"Estimating Prediction Intervals for Artificial neural Networks"*, Department of Computer and information Science, University of Pennsylvania, Philadelphia, PA 19104.
[60] Heskes, T. 1997. *"Practical confidence and prediction intervals"*, in: M.Mozer, M.Jordan & T.Petsche, eds, Advances in Neural Information Processing Systems 9, MIT Press, Cambridge, MA, pp 176-182.

$$\hat{\mu}_{y,Boot}(\boldsymbol{X}) \pm t_{(\alpha/2)B}\left(\widehat{SE}_{Boot}(f(\boldsymbol{X};\boldsymbol{\Omega})) + \chi(\boldsymbol{X})\right) \tag{3.13}$$

The prediction interval (3.13) offers the advantage that it allows for the uncertainty of the regression function as well as that of the noise. In addition, it does not rely on the assumption that the network is an unbiased estimator of the conditional mean of the target value (i.e. that the error due to bias is negligible compared with the error due to variance).

Nix and Weigand (1995)[61] have proposed a novel method of computing prediction intervals for neural nets. The method uses a NN with two output nodes, one for $\hat{\mu}_y(\boldsymbol{x};\hat{\boldsymbol{\Omega}})$, the predicted value, and a second for $\hat{\sigma}_y^2(\boldsymbol{x};\hat{\boldsymbol{U}})$, the variance of the predicted value. The network has a non-standard structure, with a second hidden layer for $\hat{\sigma}_y^2(\boldsymbol{x};\hat{\boldsymbol{U}})$ receiving inputs from both the hidden layer for $\hat{\mu}_y(\boldsymbol{x};\hat{\boldsymbol{\Omega}})$ and from the input layer. A negative log likelihood cost function is used, modified by inclusion of the input-dependent variance term $\hat{\sigma}_y^2(\boldsymbol{x}_i)$. Usually $\hat{\sigma}^2(\boldsymbol{X})$ is assumed constant and drops out after differentiation. A linear activation function is specified for the $\hat{\mu}_y(\boldsymbol{x};\hat{\boldsymbol{\Omega}})$ output unit. To ensure only positive outputs, an exponential activation function is specified for the $\hat{\sigma}_y^2(\boldsymbol{x};\hat{\boldsymbol{U}})$ output unit. A hyperbolic tangent activation function is used for the hidden layer units. Using these activation functions, differentiating the cost function with respect to the network weights gives weight update equations containing terms $1/\hat{\sigma}_y^2(\boldsymbol{x})$, which act as a form of weighted regression. An improved fit in low noise regions of the input space is claimed. The outputs obtained from this network are equivalent to training a separate network for $\hat{\sigma}_y^2(\boldsymbol{x};\hat{\boldsymbol{U}})$ using the squared residuals from $\hat{\mu}_y(\boldsymbol{x};\hat{\boldsymbol{\Omega}})$ as targets.

The Nix-Weigend network requires a three phase training process. In *Phase I* the network is trained on a training set $\boldsymbol{A}$ for the output $\hat{\mu}_y(\boldsymbol{x};\hat{\boldsymbol{\Omega}})$. This is equivalent to normal network training using a sum of squares cost function with early stopping to prevent over fitting. In *Phase II*, the weights trained in Phase I are frozen and the second hidden layer for $\hat{\sigma}_y^2(\boldsymbol{x};\hat{\boldsymbol{U}})$ added. The squared residuals from the Phase I

[61] Nix, D.A., Weigend, A.S. 1995. *"Learning Local Error Bars for Non-Linear Regression"*, In: Proceedings of NIPS 7, pp 489-496.

model are used as targets for the second output node $\hat{\sigma}_y^2(\boldsymbol{x};\hat{\boldsymbol{U}})$. The network is now trained for the output $\hat{\sigma}_y^2(\boldsymbol{x};\hat{\boldsymbol{U}})$ using the validation set $\boldsymbol{B}$ from Phase I as the training set, with set $\boldsymbol{A}$ as the validation set. In *Phase III (weighted regression)*, the available data are re-split into a new training set $\boldsymbol{A'}$ and validation set $\boldsymbol{B'}$. All network weights are unfrozen, and the network is re-trained for both output nodes on training set $\boldsymbol{A'}$, using $\boldsymbol{B'}$ as the validation set. Training is now considered complete.

The Nix-Weigend method can provide prediction intervals without bootstrap resampling or use of the Hessian matrix. Improved performance in low noise regions of the input space is claimed, due to the use of a form of weighted regression. However, the use of weighted regression means the standard errors obtained do not have their usual interpretation. This is because inference for NNs rests on the assumption the network performs a NLLS regression. In the case of weighted regression a penalty term is added to the error function and this is not the case. Weighted regression also introduces local minima in the error surface, complicating learning[62]. Moreover, since the Nix-Weigend method requires use of a special non-standard architecture, and other features specific to neural networks, it is not a generally applicable method. In tests using both synthetic data with added non-uniform Gaussian noise, and real-world data with uniform non-Gaussian noise, compared with the use of a separate network to estimate the variance, as proposed by Satchwell (1994)[63], or used by Heskes (1997) the Nix-Weigend method produces improved prediction intervals.

3.3.2 Limitations of Existing Methods

Existing methods for computing standard errors, confidence, and prediction intervals for neural nets are of three types. First, the delta method, sandwich method, and their variants, which use the Hessian matrix of second order partial derivatives of the cost function with respect to the weights and biases. Second, methods using bootstrap resampling. Third, methods which rely on directly modelling the noise. Empirical tests suggest that methods of the first type do not perform as well as methods of the second type, at least for small samples. Moreover, it is not always possible to use the delta and sandwich methods. Where there is over-fitting the required matrix inversions are

[62] Nix and Weigend (1995), p496.
[63] Satchwell, C. 1994. *"Neural Networks for Stochastic Problems: More than One Outcome for the Input Space"*, In: NCAF Conference, Aston University, September 1994.

unstable and may fail, making the necessary computations impossible. Methods based on bootstrap resampling are reported by Tibshirani (1996) to give estimates that are more accurate. While the bootstrap can provide confidence intervals for the true regression function, on its own it cannot provide prediction intervals where the target variable is unknown. Heskes (1997) has sought to overcome this limitation by proposing a method that uses a separate neural network to model the noise, in conjunction with bootstrap resampling. However, the naïve bootstrap does not provide heteroskedasticity consistent standard errors[64]. For these, the use of a more complex wild bootstrap is required. The method proposed by Nix and Weigand (1995) is of the third type. It does not have any of the above limitations, and is less computationally costly. However, it relies on a form of weighted regression. Consequently, the validity of inference using the estimated standard errors is questionable. Moreover, the method uses a non-standard topology, and features specific to the MLP, and is not generally applicable.

3.4 Robust Practical Prediction Intervals and PR

A new method for obtaining standard errors, confidence, and prediction intervals that is applicable not only to MLPs, but to *any* non-linear regression method of comparable generality is now described. The method is robust to heteroskedasticity and practical to implement. It allows the standard error to be obtained directly and avoids the bootstrapping that is otherwise a practical necessity in obtaining confidence intervals for the true regression.

The method is based on a network with two outputs; one fitted to the target variable and the other to its squared error. It differs from the Nix and Weigend (1995) method because: a) It deploys a standard NN architecture and b) It uses a sum of squares cost function and does not assume a Gaussian noise distribution. c) It uses a training algorithm with independent training and validation sets, rather than interchanging validation sets. This latter feature is due to a suggestion by Heskes (1997), that it is desirable that the training set for fitting the squared errors, is disjoint from either the training or validation sets used for fitting the target variable. The reason is, when a network is trained using early stopping, training is stopped when the sum of squared

[64] Wu, C.J.F. 1986. "Jacknife Bootstrap and other Resampling Methods in Regression Analysis", Annals of Statistics 14, pp1261-1295.

errors is minimized not on the training set, but on a separate validation set. Thus, the model obtained is a function of *both* data sets, and the validation set cannot be viewed as independent for the purpose of fitting the squared errors. The theoretical basis of the method follows.

3.4.1 Least Squares Derivation

The object in training (fitting) a MLP is not to memorise features specific to the training set, but to model the underlying data generating process, so that when presented with a new input vector $\boldsymbol{x}$, the trained network gives the best possible estimate of the target. The most comprehensive description of the DGP is a statistical one, in terms of the joint probability density $P(\boldsymbol{x},\boldsymbol{d})$ of the input vector $\boldsymbol{x}$ and the target vector $\boldsymbol{d}$. This density can be expressed as the product of the conditional distribution $P(\boldsymbol{d} \mid \boldsymbol{x})$ of the target vector $\boldsymbol{d}$ conditioned on the input vector $\boldsymbol{x}$, and the unconditional distribution $P(\boldsymbol{x})$ of the input vector

$$P(\boldsymbol{x},\boldsymbol{d}) = P(\boldsymbol{d} \mid \boldsymbol{x})P(\boldsymbol{x}) \tag{3.14}$$

where

$$P(\boldsymbol{x}) = \int P(\boldsymbol{x},\boldsymbol{d})d\boldsymbol{d} \tag{3.15}$$

An MLP trained by minimising a sum-of-squares error defined over a training set, approximates the means of the elements of a target vector $\boldsymbol{d}$ conditioned on a corresponding input vector $\boldsymbol{x}$. The optimisation results in estimation of a vector $\hat{\boldsymbol{\Omega}}$ of weights and biases that minimises the cost function. The function to be minimised takes the form

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}[\mathrm{f}_j(\boldsymbol{x}_i;\hat{\boldsymbol{\Omega}}) - d_{ij}]^2 \tag{3.16}$$

In equation (3.16) d_{ij} is the jth element of the ith target vector. $\mathrm{f}_j(\boldsymbol{x}_i;\hat{\boldsymbol{\Omega}})$ is the corresponding network estimate. Asymptotically as n, the size of the data set, tends to infinity and assuming the function $\mathrm{f}_j(\boldsymbol{x}_i;\hat{\boldsymbol{\Omega}})$ has sufficient flexibility (i.e. degrees of freedom), bias and variance tend to zero yielding the optimum least squares solution.

In the limit, the summation over n in (3.16) becomes an integral over the joint probability density[65]

$$C^{LS} = \lim_{n\to\infty} \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{m}[\mathrm{f}_j(\boldsymbol{x}_i,\hat{\boldsymbol{\Omega}}) - d_{ij}]^2 \tag{3.17}$$

$$= \frac{1}{2}\sum_{j=1}^{m}\iint[\mathrm{f}_j(\boldsymbol{x},\hat{\boldsymbol{\Omega}}) - d_j]^2 P(d_j \mid \boldsymbol{x})P(\boldsymbol{x})\, dd_j\, d\boldsymbol{x} \tag{3.18}$$

where $1/n$ in (3.17) is a convergence factor. The cost function can be minimised using functional differentiation with respect to $\mathrm{f}_j(\boldsymbol{x},\hat{\boldsymbol{\Omega}})$ and setting the derivative to zero.

$$\frac{\delta C^{LS}}{\delta \mathrm{f}_j(\boldsymbol{x},\hat{\boldsymbol{\Omega}})} = 0 \tag{3.19}$$

Substituting (3.18) into (3.19) and using (3.14) yields the following solution for the minimising function

$$\mathrm{f}_j(\boldsymbol{x},\hat{\boldsymbol{\Omega}}) = E(d_j \mid \boldsymbol{x}) = d_j(\boldsymbol{x}) \tag{3.20}$$

Thus, the output of the network function corresponds to the conditional means of the elements of the target vector ***d***, conditioned on the input vector ***x***. The result (3.20) depends only on the generality of the non-linear mapping represented by the network function. It does not specifically require use of a MLP and thus extends to any comparable non-linear mapping of sufficient flexibility.

The (global) conditional variance corresponding to the conditional mean (3.20) is given by

$$Var(d_j \mid \boldsymbol{X}) = \frac{\sum_{i=1}^{n} e_{ij}^2}{(n-k-1)} \tag{3.21}$$

where $\boldsymbol{X}$ is the matrix of input data, e_{ij}^2 is the squared residuals for $d_j(\boldsymbol{x}_i)$ at the minimum of the cost function, n is the number of observations in the data set, and k is the applicable degrees of freedom. Given (3.21)the conditional distribution of the target $P(d_j \mid \boldsymbol{x}_i)$ is characterised by a two parameter distribution with a mean given by $\mathrm{f}_j(\boldsymbol{x},\hat{\boldsymbol{\Omega}})$ and a (global) variance given by $Var(d_j \mid \boldsymbol{X})$. However, the use of a least squares cost function does *not* require the assumption that this distribution is Gaussian.

[65] Bishop (1997), p201.

Suppose, that as well as d_j the target vector $\boldsymbol{d}$ has an additional element $\sigma_j^2(\boldsymbol{x})$ where $\sigma_j^2(\boldsymbol{x}) = \{d_j - E(d_j \mid \boldsymbol{x})\}^2$, the squared residuals from the network estimate of $d_j(\boldsymbol{x})$. Then it follows from (3.20) that

$$\sigma_j^2(\boldsymbol{x}) = E[\{d_j - E(d_j \mid \boldsymbol{x})\}^2 \mid \boldsymbol{x}] = Var(d_j \mid \boldsymbol{x}) \tag{3.22}$$

The function $\hat{\sigma}_j^2(\boldsymbol{x})$ is modelled by adding an additional output node to the MLP trained to fit the squared residuals of $\hat{d}_j(\boldsymbol{x})$. Using (3.22) in place of (3.21) facilitates estimation of a separate variance parameter for each target d_j conditioned on the corresponding input vector $\boldsymbol{x}$, and is equivalent to using White's heteroskedasticity consistent estimator[66].

3.4.2 Maximum Likelihood Derivation

If the conditional distribution of the target data is assumed to be Gaussian, the result (3.20) can be derived using maximum likelihood[67]. Under the Gaussian assumption $P(d_j \mid \boldsymbol{x})$ can be written as

$$P(d_j \mid \boldsymbol{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{[\mathrm{f}_j(\boldsymbol{x};\hat{\boldsymbol{\Omega}}) - d_j]^2}{2\sigma^2} \right) \tag{3.23}$$

where σ^2 is a global variance parameter that can be estimated by (3.21). Again, this is easily extended to obtain a more general distributional assumption by substituting (3.22) for (3.21) in equation (3.23) giving.

$$P(d_j \mid \boldsymbol{x}) = \frac{1}{(2\pi)^{1/2}\sigma_j(\boldsymbol{x})} \exp\left(-\frac{[\mathrm{f}_j(\boldsymbol{x};\hat{\boldsymbol{\Omega}}) - d_j]^2}{2\sigma_j^2(\boldsymbol{x})} \right) \tag{3.24}$$

Maximising the likelihood is equivalent to minimising the negative log likelihood. Forming the negative log likelihood of (3.24) and omitting constants gives

$$C^{-LL} = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(Ln\ \sigma_j(\boldsymbol{x}_i) + \frac{[\mathrm{f}_j(\boldsymbol{x}_i,\hat{\boldsymbol{\Omega}}) - d_{ij}]^2}{2\sigma_j^2(\boldsymbol{x}_i)} \right) \tag{3.25}$$

Taking the limit as before gives the integral form

$$C^{-LL} = \sum_{j=1}^{m}\iint\left(Ln\ \sigma_j(\boldsymbol{x}) + \frac{[\mathrm{f}_j(\boldsymbol{x},\hat{\boldsymbol{\Omega}}) - d_j]^2}{2\sigma_j^2(\boldsymbol{x})} \right) P(d_j \mid \boldsymbol{x})P(\boldsymbol{x})\, dd_j\, d\boldsymbol{x} \tag{3.26}$$

---

[66] Thomas (1997), p 290.

[67] Bishop, C.M. 1994. *"Mixture Density Networks"*, Technical report NCRG/4288, Neural Computing Research Group, Aston University.

Functional differentiation is again used to minimise the errors for the network outputs for the mean and variance functions. For the mean

$$\frac{\delta C^{-LL}}{\delta \mathrm{f}_j(\boldsymbol{x},\hat{\boldsymbol{\Omega}})} = 0 = P(\boldsymbol{x}) \int \frac{[\mathrm{f}_j(\boldsymbol{x},\hat{\boldsymbol{\Omega}}) - d_j]}{\sigma_j^2(\boldsymbol{x})} P(d_j \mid \boldsymbol{x})\, dd_j \tag{3.27}$$

Rearranging and simplifying (3.27) gives the standard result of equation (3.20). For the variance, (3.26) is minimised with respect to the function $\sigma_j(\boldsymbol{x})$ giving

$$\frac{\delta C^{-LL}}{\delta \sigma_j(\boldsymbol{x})} = 0 = P(\boldsymbol{x}) \int \left(\frac{1}{\sigma_j(\boldsymbol{x})} - \frac{[\mathrm{f}_j(\boldsymbol{x},\hat{\boldsymbol{\Omega}}) - d_j]^2}{\sigma_j^2(\boldsymbol{x})^3} \right) P(d_j \mid \boldsymbol{x})\, dd_j \tag{3.28}$$

Using (3.20) again and solving for $\sigma_j^2(\boldsymbol{x})$ gives (3.22). This approach is based on maximum likelihood and gives a biased estimate of the variance, because it makes use of an estimated mean, rather than the (unknown) true mean. The relationship between the true variance and its estimate obtained under maximum likelihood is given by

$$\hat{\sigma}^2 = \frac{n-k-1}{n}\sigma^2, \text{ hence } \sigma^2 = \frac{n}{n-k-1}\hat{\sigma}^2 \tag{3.29}$$

where k is the appropriate degrees of freedom[68].

Thus, it is shown here that a MLP with two output nodes, the first trained to fit a target value, and the second trained to fit the squared residuals of the first fit, can produce an estimate of the mean and variance of the conditional distribution of the target in both the least squares and maximum likelihood frameworks. The maximum likelihood derivation requires the assumption that the conditional distribution of the target data is Gaussian. This assumption is not made for the proposed method. However, maximum likelihood and least squares estimators are otherwise equivalent. Moreover, the result (3.20) which is central to the proposed method requires only the use of a least squares cost function, and a sufficiently flexible form of non-linear regression.

3.4.3 Derivation of the *PR* Criterion

The prediction risk is defined as the expected performance of an estimator on future data, and is the expected value of the mean squared error $E[MSE]$. In the infinite data limit the *PR* is given by

$$PR = \int \left(p(\boldsymbol{x})[\mathrm{f}(\boldsymbol{x};\boldsymbol{\Omega}) - \mathrm{f}(\boldsymbol{x};\hat{\boldsymbol{\Omega}})]^2 + \sigma_\varepsilon^2 \right) d\boldsymbol{x} \tag{3.30}$$

[68] Equation (3.29) also applies to least squares estimators. The value of k depends on the particular regression technique used.

where σ_ε^2 in (3.30) is the (unknown) noise variance, $p(\boldsymbol{x})$ is the unconditional distribution of $\boldsymbol{x}$, and the remaining term is the squared error of the regression. The prediction risk can also be defined for other cost functions but the squared error is used here. For finite data sets (3.30) can be approximated by

$$\widehat{PR} = E\left(\frac{1}{N}\sum_{i=1}^{N}[\tilde{d}_i - \mathrm{f}(\tilde{\boldsymbol{x}}_i, \hat{\boldsymbol{\Omega}})]^2\right) \tag{3.31}$$

In equation (3.31) $(\tilde{\boldsymbol{x}}_i, \tilde{d}_i)$ are new observations, not used in training or validating the network, and N is the total number of such observations. Note that the summation in (3.31) is over the *expected* squared error $[\tilde{d}_i - \mathrm{f}(\tilde{\boldsymbol{x}}_i, \hat{\boldsymbol{\Omega}})]^2$. This spans the squared error of the regression, and the noise variance, of equation (3.30). From equation (3.20)

$$\mathrm{f}_j(\boldsymbol{x}, \hat{\boldsymbol{\Omega}}) = E(d_j \mid \boldsymbol{x}) = d_j(\boldsymbol{x})$$

also

$$\sigma_j^2(\boldsymbol{x}) = E[\{d_j - E(d_j \mid \boldsymbol{x})\}^2 \mid \boldsymbol{x}] = Var(d_j \mid \boldsymbol{x})$$

so that (3.22) can be written as

$$\sigma_j^2(\boldsymbol{x}) = E[\{d_j - \mathrm{f}_j(\boldsymbol{x}, \hat{\boldsymbol{\Omega}})\}^2 \mid \boldsymbol{x}] = Var(d_j \mid \boldsymbol{x}) \tag{3.32}$$

Thus, $\sigma_j^2(\boldsymbol{x})$ is nothing other than the expected squared error for d_j given some input vector $\boldsymbol{x}$. Equation (3.31) can be rewritten as

$$\widehat{PR}_j = E\left(\frac{1}{N}\sum_{i=1}^{N}\hat{\sigma}_j^2(\boldsymbol{x}_i)\right) \tag{3.33}$$

Equation (3.31) can only be used to estimate the prediction risk for test sets where d_j is known, (so-called test set validation). Test set validation is only reliable where the test set is large[69]. Otherwise, *PR* must be estimated using resampling techniques such as cross-validation, which is computationally costly. By contrast, the *PR* estimator (3.33) can be used to estimate the prediction risk for new inputs $\boldsymbol{x}$, for which there is no corresponding target d. Moreover, the method of estimation does not require large test sets, or use of resampling techniques.

[69] Moody (1994), section 3.1.

3.5 Practical Implementation and Tests

Practical application of the proposed method requires the use of a special training algorithm. The algorithm given in Table 6 is the result of extensive investigations to find an effective practical implementation of the theory presented in section 3.4.1.

Table 6 Training Algorithm

Phase	Description
I	a) Randomly split the training data into two data sets, Set A and Set B. b) USING SET A, train a Phase I NN to fit the target variable $d(\boldsymbol{x})$. c) Run the trained NN model ON SET B, create a set of squared residuals. (1)
II	a) USING SET B, train a Phase II NN with two output nodes. Use the variable $d(\boldsymbol{x})$ as the target for the FIRST output node. Use the squared residuals created in Phase I in SET B as the target for the SECOND output node.
III (optional)	a) Run the Phase II model on SET A, create a set of squared residuals for the target $d(\boldsymbol{x})$. b) USING SET A, train a Phase III NN with two output nodes. Use the variable $d(\boldsymbol{x})$ as the target for the first output node. Use the squared residuals created in step III a) in SET A as the target for the second output node. (2)
Notes	(1) By using squared residuals on a test set (Set B) as the second target for Phase II, over fitting and consequent underestimation of the standard error is avoided. (2) Phase III may give improved results on certain data, but generally produces inferior results to Phase II, and may thus be omitted. For training, Set A is itself randomly split into a training and a validation portion; as is Set B. Testing of each Phase is performed on an independent test set, Set C.

3.5.1 Testing the Method

The following sequence of tests was performed. First the method was tested as Example#1, using the same synthetic data, and single input variable used by Nix and Weigend (1995). This allowed a performance comparison with their method. Next the method was tested as Example#2, using options market data. Synthetic option prices and noise generated by known underlying functions were used. Example#2 is deliberately restricted to two input variables, moneyness, and maturity $[m,t]$, following Hutchinson et al (1994). This is to allow the use of a known smooth noise variance function. The purpose of Example#2, was to test the method in a more realistic multi dimensional, option pricing context, while retaining comparability with Example#1. The method was next tested using actual option prices, as Example#3. The same options market data as used for Example#2 was again used. For comparability with Example#2 the model was again restricted to the two inputs $[m,t]$. The underlying true regression function and noise variance function were unknown for Example#3 however, in contrast to Example#1 and #2 where synthetic prices and noise were used, generated by known functions. Finally, the method was tested as Example#4 using actual prices, and the five BS inputs$[S,X,t,r,IV]$ as defined in Appendix D. The same data sets were

again used. This allowed comparison with the benchmark BS formula on the one hand, and the preceding Example#3 on the other.

The sequence of examples features increasing dimensionality from the one dimensional Example#1, through the two dimensional Example#2 and #3, to the five dimensional Example#4. The sequence also relaxes the condition of synthetic inputs, target data, and noise, used for Example#1. It moves through the synthetic target and noise, but actual market inputs of Example#2, to the actual prices, market inputs, and residuals of Example#3 and #4. At the beginning of the sequence, a direct comparison was possible with the method of Nix-weigend, and at the end, a direct comparison with the benchmark BS formula was possible. The remainder of section 3.5 of the thesis gives the details and test results for Examples#1 to #4.

3.5.2 Tests Using Standard Synthetic Data

Nix and Weigend (1995) defined a univariate synthetic example to demonstrate the effectiveness of their model. For comparison purposes the proposed training algorithm and network were applied to the same univariate synthetic data, and called Example #1 in their paper and here. This example used a one-dimensional data set where $y(\boldsymbol{x})$ the true regression, and $\sigma^2(\boldsymbol{x})$, the variance of the noise, were known. The true regression $y(\boldsymbol{x})$, is given by the equation $y(\boldsymbol{x}) = Sin(3x)Sin(5x)$, where x is a uniformly distributed random number from the interval [0, π/2]. The noise $n(\boldsymbol{x})$ consists of numbers from the normal distribution $N[0, \sigma^2(\boldsymbol{x})]$, where $\sigma^2(\boldsymbol{x}) = 0.02 + 0.25[1 - Sin(5x)]^2$ [70]. The target value for training is $d(\boldsymbol{x}) = y(\boldsymbol{x}) + n(\boldsymbol{x})$.

The following procedure was adopted. For Phase I, a network with a single input node, 10 hidden layer nodes and a single output node to fit $d(\boldsymbol{x})$ was used. Phases II and III used a network with a single input node, 20 hidden layer nodes and 2 output nodes, one to fit $d(\boldsymbol{x})$ and one to fit $\sigma^2(\boldsymbol{x})$. The numbers of hidden nodes used were the same as in the Nix-Weigend method, except that here they were fully connected, in a single layer. Fig. 9 shows a plot of the data points, the true regression $y(\boldsymbol{x})$, and the approximate prediction band, for the Phase III model, obtained on a test set. The effect of Phase III was to improve the Adj. R^2 figure, and the F-statistic, for the estimated noise variance

[70] Heskes (1997) used similar trigonometric functions for the true regression and the noise variance.

function. Table 7 shows the results of statistical tests for Example#1 in predicting the true regression, the true noise variance function, the target data points $d(\boldsymbol{x})$, and the actual squared residuals.

Fig. 9 Prediction Bands for Example#1

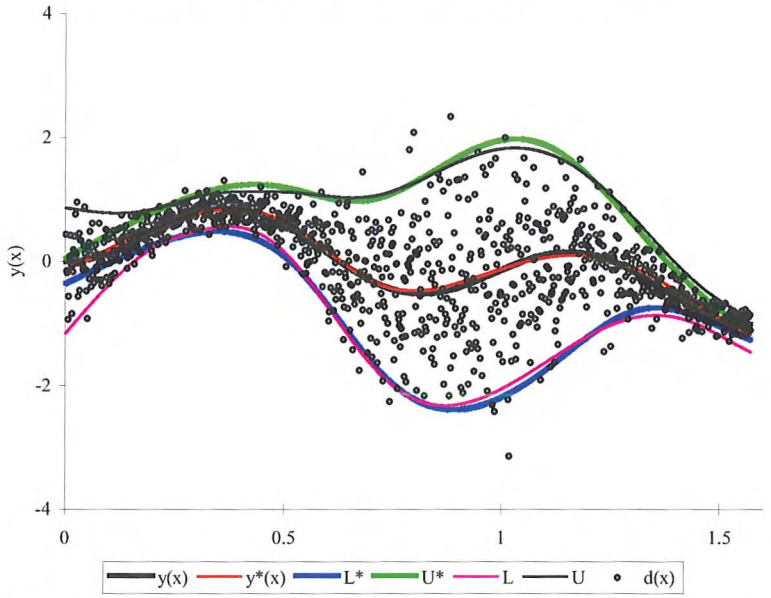


Fig.9 Here, $y(\boldsymbol{x})$ is the true regression. $d(\boldsymbol{x})$ are the target data points. $y^*(x)$ is the estimate of the true regression. L and U are the true lower and upper prediction intervals and L* and U* are the estimated prediction intervals obtained using $\sigma^{*2}(\boldsymbol{x})$, the network estimate of the noise variance function.

In Table 7 $\mu_y^*(x)$ is the estimate produced by the node having $d(\boldsymbol{x})$ as target, and $\sigma^{*2}(\boldsymbol{x})$ is the estimate produced by the node having squared residuals as target. In the comparison of the network estimates with the true regression function and noise variance function, unbiased estimates of the mean of the true regression $y(\boldsymbol{x})$, and the true noise variance function $\sigma^2(\boldsymbol{x})$ were obtained for both Phase II and Phase III. For Phase III, the F-statistics suggest the distributions of values for the true regression and true noise variance functions, were also well recovered. Approximate upper and lower prediction intervals based on the estimated noise variance function were also unbiased estimates of the true upper and lower prediction intervals. The fit to the true regression function is very good in Phase II and III with R^2 and Adj. R^2 figures > 0.99. The fit to the noise variance function was also excellent with Phase II and III R^2 and Adj. R^2 figures >0.94 in all cases. In the comparisons of the network estimates with the actual target $d(\boldsymbol{x})$, and the actual squared residuals, the much poorer R^2, Adj. R^2, and F-statistic figures are consistent with the noisy, scattered, target data points. However, in both Phases II and III, the t-test results indicate unbiased estimates of the mean of the target d. In Phase III, the estimate of the mean of the squared residuals is also unbiased.

Table 7 Estimates by Proposed Network (Example#1)

Phase	Layers	Inputs	Outputs	R^2	Adj.R^2	F-stat		t-statistic		t-test
	[nodes]	x				F_{crit}(1 tail)	F_{calc}(0.05)	t_{crit}(2 tail)	t_{calc}(0.05)	[bias]
Comparison with True Regression Function										
I	1-10-1	x	$\mu_y^*(x)$	0.561	0.561	1.03	1.75	1.96	0.28	Unbiased
Comparison with True Regression Function & Noise Variance Function										
II	1-20-2	x	$\mu_y^*(x)$	0.994	0.994	1.03	1.01	1.96	0.95	Unbiased
II	1-20-2	x	$\sigma^{*2}(x)$	0.965	0.944	1.03	1.74	1.96	1.64	Unbiased
III	1-20-2	x	$\mu_y^*(x)$	0.992	0.991	1.03	1.01	1.96	-1.12	Unbiased
III	1-20-2	x	$\sigma^{*2}(x)$	0.987	0.989	0.97	0.86	1.96	-1.96	Unbiased
Comparison with Actual Target (d)										
I	1-10-1	x	$\mu_y^*(x)$	0.234	0.234	1.03	4.18	1.96	0.15	Unbiased
Comparison with Actual Target (d) & Squared Residuals										
II	1-20-2	x	$\mu_y^*(x)$	0.423	0.423	1.03	2.41	1.96	0.68	Unbiased
II	1-20-2	x	$\sigma^{*2}(x)$	0.211	0.967	1.03	4.41	1.96	-8.55	Biased
III	1-20-2	x	$\mu_y^*(x)$	0.422	0.422	1.03	2.4	1.96	-0.91	Unbiased
III	1-20-2	x	$\sigma^{*2}(x)$	0.338	0.337	1.03	4.38	1.96	0.32	Unbiased

Table 7 Shows the proposed network produces unbiased estimates $\mu_y^*(x)$, of the underlying true regression function, and $\sigma^{*2}(x)$ of the noise variance function. Phase III estimates of the actual target d and actual squared residuals are also unbiased. The t-test for the means shows no difference at the 95% level.

For comparison of the proposed network performance with that of Nix-Weigend, the statistics used by Nix and Weigend (1995) were computed. These are shown in Table 8 for Example#1 and Table 9 for Example#2. The two tables are shown together, however the Table 9 results are discussed in section 3.5.3. Table 8 shows that compared to the Nix-Weigend network there was little improvement in the fit to the target *d(x)* between Phases II and III. However, the Phase III fit (row 3) for the proposed network was close to the best attainable, deviating only by 0.6%. The Nix-weigend network does not approach the best attainable figure quite so closely, deviating by 1.4%. The proposed network figures for correlation of the actual absolute errors with the network prediction and the true values (rows 5 and 6), improved slightly on the corresponding figures for the Nix-Weigend network, even in Phase II. The distributions of errors reported in rows 7 and 8 differed only slightly from those of Nix-Weigend. Overall, the results in Table 8 show the proposed network performed comparably with a Nix-Weigend network and outperformed it slightly on the errors and correlations. However, the results in Table 8 are not based on hypothesis tests or confidence intervals, therefore the results presented in Table 7 are preferred. The results presented in Table 7 showed that the proposed network was able to provide unbiased estimates of an underlying true regression function, an associated noise variance function, and the actual targets and squared errors in the univariate case.

Table 8 Results for Example#1: Methods Compared

		This Work		Nix-Weigend	
row		Test Set	($n=10^4$)	Test Set	($n=10^5$)
	Target d	E_{NMS}	Our Mean Cost	E_{NMS}	NW Mean Cost
1	Phase I	0.764	0.454	0.593	0.882
2	Phase II	0.577	0.344	0.593	0.566
3	Phase III	0.578	0.344	0.570	0.462
4	n(x) (exact additive noise)	0.575	0.343	0.563	0.441
	Target e^2	ρ (PIII)	ρ (PII)	ρ (PIII)	ρ (PII)
5	ρ(σ*(x), residual errors)	0.571	0.569	0.548	N/a
6	ρ(σ(x), residual errors)	0.586	0.585	0.584	N/a
	Distribution P(III)	1 std.	2 std.	1 std.	2 std.
7	% of errors < σ*(x); 2σ*(x)	67.4	93.1	67.0	94.6
8	% of errors < σ(x); 2σ(x)	66.9	95.0	68.4	95.4
9	(exact Gaussian)	68.3	95.4	68.3	95.4

Table 8 E_{NMS} is the mean squared error normalised by the (global) variance of the target d. The Mean Cost is the mean of the cost function $(d-d^*)^2$. Row 4 gives these figures for $[(d-y(x))^2 = n(x)^2]$ and represents the best performance attainable. Row 5 gives the correlations between the absolute errors and the network estimate for the standard deviation of the errors. Row 6 gives the correlations between the absolute errors and the true noise standard deviation. Row 7 gives the percentage of absolute errors that are less than 1 and 2 times the corresponding network estimate for the standard deviation of the error. Row 8 gives the percentage of absolute errors that are less than 1 and 2 times the corresponding true noise standard deviation. Row 9, which is included for comparison purposes, gives the percentage of observations that are less than 1 and 2 standard deviations in a Gaussian distribution.

Table 9 Results for Example#2: Synthetic Option Prices + Noise

row		**Test Set**	
	Target d = C_{NN} + noise	E_{NMS}	Our Mean Cost
1	Phase I	0.059	564.6
2	Phase II	0.062	599.4
3	Phase III	0.059	566.8
4	n(x) (exact additive noise)	0.058	556.9
	Target e^2	ρ (PIII)	ρ (PII)
5	ρ(σ*(x), residual errors)	0.537	0.536
6	ρ(σ(x), residual errors)	0.562	0.591
	Distribution (PII)	1 std.	2 std.
7	% of errors < σ*(x); 2σ*(x)	51.3%	81.4%
8	% of errors < σ(x); 2σ(x)	70.0%	96.1%
	Distribution (PIII)		
9	% of errors < σ*(x); 2σ*(x)	32.5%	60.2%
10	% of errors < σ(x); 2σ(x)	71.4%	96.2%
11	(exact Gaussian)	68.3%	95.4%

Table 9 Rows 1 to 8 are as Table 8. Rows 9 and 10 for PhaseIII correspond to rows 7 and 8 for Phase II. Row 11, corresponds to row 9 in Table 8. Here **(x)** represents the vector of input variables $[t,m]$.

3.5.3 Tests Using Synthetic Option Prices

This section reports results for Example#2 where the proposed network was applied in the more realistic multivariate setting of option pricing. The method was tested using synthetic option prices and synthetic noise. The option market data used were from LIFFE. They consisted of daily closing prices for the FTSE-100 index call option for all trading dates from 13 March 1992 to 1 April 1997. The raw data set contained 119,413 records. The procedures used for cleaning and preparing the data are described in section 4.2.1. The cleaned data set comprised 14,254 records. This data set was randomly split into a training set and a test set. The resulting training sets contained 7,083 records with 3629 in Set A and 3454 in Set B. 50% of these were randomly sampled and used for validation. The test set contained 7,171 records.

The synthetic option prices were created using a trained neural net option pricing model as the underlying known true regression function. For this purpose, the approach of Hutchinson et al (1994) was followed, and the volatility and risk-free interest rate were omitted as inputs. The neural net was trained using observed market prices as the target and the variables moneyness, ($m = S/X$), and time to maturity, t, as inputs; S represents the price of the asset in index points and X is the strike price for the option. Analysis of squared residual errors for neural net option pricing models indicated that $\sigma^2(t,m) = 510t^4 + 361m^{17}$ was an acceptable parameterization for the known true noise variance function for approximating the underlying residuals; it is important to emphasize that this function has no significance other than providing a noise variance model for this example. Using $\sigma^2(t,m)$, a synthetic noise distribution very similar to that for real residual errors for neural net option pricing models was obtained, as indicated by variance, standard deviation, skewness, and kurtosis. Synthetic noise from the normal distribution N was drawn as $N(0,\sigma^2(t,m))$ and added to the outputs of the trained NN to generate a synthetic target option price $d(t,m)$. The obtained target $d(t,m)$, was not significantly different from observed market prices in t-tests and F-tests. As in Example #1, the aim was to determine whether the method could successfully recover an underlying known regression function and noise variance function. The results for Example#2 are presented in Table 10.

Table 10 Estimates by Proposed Network (Example#2)

Phase	Layers	Input	Outputs	R^2	Adj.R^2	F_Statistic		t-statistic		t-test
	[nodes]	$x=[t,m]$				F_{crit}(1 tail)	F_{calc}(0.05)	t_{crit}(2 tail)	t_{calc}(0.05)	[bias]
Comparison with True Regression Function										
I	2-10-1	x	$\mu_y^*(x)$	0.997	0.998	1.04	1.08	1.96	-0.02	Unbiased
Comparison with True Regression Function & Noise Variance Function										
II	2-20-2	x	$\mu_y^*(x)$	0.995	0.997	0.96	0.98	1.96	-1.62	Unbiased
II	2-20-2	x	$\sigma^{*2}(x)$	0.845	0.561	0.96	0.45	1.96	1.52	Unbiased
III	2-20-2	x	$\mu_y^*(x)$	0.997	0.998	1.04	1.08	1.96	-0.61	Unbiased
III	2-20-2	x	$\sigma^{*2}(x)$	0.900	0.856	0.96	0.75	1.96	6.20	Biased
Comparison with Actual Target (d)										
I	2-10-1	x	$\mu_y^*(x)$	0.942	0.967	1.04	1.11	1.96	0.49	Unbiased
Comparison with Actual Target (d) & Squared Residuals										
II	2-20-2	x	$\mu_y^*(x)$	0.939	0.965	1.04	1.00	1.96	-1.11	Unbiased
II	2-20-2	x	$\sigma^{*2}(x)$	0.246	-0.743	1.04	1.29	1.96	1.16	Unbiased
III	2-20-2	x	$\mu_y^*(x)$	0.942	0.967	1.04	1.11	1.96	-0.10	Unbiased
III	2-20-2	x	$\sigma^{*2}(x)$	0.219	-0.627	1.04	1.51	1.96	4.39	Biased

Table 10 shows the proposed network produces an unbiased estimate, $\mu_y^*(x)$, of both the underlying true regression function and the target d in all three training phases. The estimate $\sigma^{*2}(x)$ is also an unbiased estimate of both the true noise variance function and actual squared error for PhaseII; the corresponding Phase III estimates are biased.

In Table 10 $\mu_y^*(x)$ is the estimate produced by the node having $d(x)$ as target, and $\sigma^{*2}(x)$ is the estimate produced by the node having the squared residuals as target. In the comparison of the network estimates with the true regression function and noise variance function, unbiased estimates of the mean of the true regression $y(x)$, and the true noise variance function $\sigma^2(x)$ were obtained for Phase II. The Phase III estimate of the mean of the noise variance function $\sigma^2(x)$ was biased. For both Phase II and III, the F-statistics suggest the distribution shapes for the true noise variance function, were again well recovered. The fits to both the true regression and noise variance function for both PhaseII and III are again very good, as measured by R^2 and Adj. R^2. In the comparisons of the Example#2 estimates with the actual target $d(x)$, and the actual squared residuals, the R^2, and Adj. R^2 figures for $\mu_y^*(x)$ are much better than the corresponding results for Example#1. This is because the synthetic option price data is far less noisy than the corresponding data used for Example#1. Again, in both Phases II and III, the t-test results indicate unbiased estimates of the mean of the target $d(t,m)$. In Phase II, the estimate of the mean of the squared residuals is also unbiased.

For comparison of the proposed network performance with that of the Nix-weigend network, the statistics used by Nix and Weigend (1995) were computed. These are shown in Table 9 in section 3.5.2. In Table 9 The Phase III fit for the target $d(t,m)$, (Row 3) is only 0.001, (1.7%) more than the lowest attainable value (Row 4). The

Phase II fit (Row 2), was not quite as good but still only 0.004 (7%) more than the lowest attainable value. Although there was no statistically significant difference at the 95% level in the t-tests, the Phase II fit to E_{NMS} was slightly poorer than the Phase I fit (Row 1). This may be because, in contrast to Nix and Weigend (1995), the Phase II used here involved training a new model constrained to fit both the target $d(t,m)$ and the squared residuals from Phase I. Like Nix and Weigend (1995) 10 hidden layer nodes per output node were used, but here these were in a single hidden layer of 20 nodes with full connectivity to all input and output nodes. Pruning runs, not reported here, indicated fewer nodes could achieve the relevant accuracy. As in Example#1 the correlation of the absolute errors with the estimated absolute errors and with the true noise standard deviation (Rows 5 and 6) improved slightly from Phase II to Phase III. The correlation results in Table 9 are of a similar order to those for Example#1 in Table 8. Row 7 and Row 9 distribution results show the dispersion of the actually occurring absolute errors was greater than indicated by the estimated and true noise standard deviation results given in Row 8 and Row 10. The decreased correlation of absolute values of residual errors with the known noise standard deviation (Row 6) suggested that Phase III training should be omitted in the more realistic multivariate setting for this data. This conclusion was supported by Table 10, where hypothesis tests for Example#2 showed that the Phase III estimate $\sigma^{*2}(\boldsymbol{x})$, was a biased estimate of both the true noise variance and the squared residuals.

The Table 9 and 10 results for PhaseII training in Example#2 showed the proposed network produced an unbiased estimate $\sigma^{*2}(\boldsymbol{x})$, of the input dependent noise variance function $\sigma^2(t,m)$, which is a smooth function of time to maturity t, and moneyness m. Moreover, $\sigma^{*2}(\boldsymbol{x})$ was also an unbiased estimate of the actually occurring residual errors for $\mu_y^*(\boldsymbol{x})$. These results suggested that given a set of unseen input variables for which there is no corresponding targets, the proposed network was capable of producing an unbiased estimate of a target $d(t,m)$, in this case synthetic option prices. An unbiased estimate of the underlying regression giving rise to the target data, and a corresponding (known) noise variance function, was also obtained. The unbiased estimate of both the mean of the target and the true noise variance function suggested that prediction intervals based upon the proposed network, provided a good estimate of the 95% prediction intervals.

3.5.4 Tests Using Actual Option Prices

Example#3 used the same data set as Example#2. However, actual observed market prices of options, corresponding to the input variables *t*, and *m*, were now used as targets in place of the synthetic prices created as targets for Example#2.

Table 11 Estimates by Proposed Network (Example#3)

Phase	Layers	Input	Output	R^2	Adj.R^2	F_Statistic		t-statistic		t-test
	[nodes]	x = [t,m]				F_{crit}(1 tail)	F_{calc}(0.05)	t_{crit}(2 tail)	t_{calc}(0.05)	[bias]
Comparison with Actual Target (O)										
I	2-10-1	x	$\mu_y^*(x)$	0.960	0.999	1.04	1.09	1.96	0.47	Unbiased
Comparison with Actual Target (O) & Squared Residuals										
II	2-20-2	x	$\mu_y^*(x)$	0.932	0.972	1.04	1.13	1.96	-0.31	Unbiased
II	2-20-2	x	$\sigma^{*2}(x)$	0.385	0.347	1.04	6.14	1.96	0.63	Unbiased
III	2-20-2	x	$\mu_y^*(x)$	0.957	0.974	1.04	1.10	1.96	0.70	Unbiased
III	2-20-2	x	$\sigma^{*2}(x)$	0.435	0.365	1.04	1.14	1.96	-6.59	Biased

Table 11 shows the proposed network produces an unbiased estimate, $\mu_y^*(x)$, of the mean of the target *O* in all three training phases. The estimate $\sigma^{*2}(x)$ is also an unbiased estimate of the actual squared error for PhaseII; the corresponding Phase III estimate is biased.

Table 11 reports the results for Example#3. In this case, the comparisons of the network estimates are with the actual target *O*, and the actual squared residuals for the estimate $\mu_y^*(x)$ only. There is no known underlying regression and true noise variance function. The pattern for Example#2 was repeated. The Example#3 estimate for the mean of the target *O* was unbiased for all three training phases with high values for R^2. and Adj. R^2. The Phase II estimate $\sigma^{*2}(x)$, was an unbiased estimate of the actual squared residuals of $\mu_y^*(x)$, as indicated by the t-test results. However the Phase III estimate was biased, like the corresponding Example#2 result. The R^2 and Adj. R^2 figures for the estimate $\sigma^{*2}(x)$ were better than the corresponding Example#2 results. These results suggested that the proposed network could produce unbiased estimates of both the mean and squared residuals of the target values, where those targets were actual observed option prices corresponding to unseen input variables. The biased estimate $\sigma^{*2}(x)$ obtained in Phase III is further evidence that Phase III training is superfluous in the more realistic setting. Moreover, the estimate $\mu_y^*(x)$ is not improved in Phase III, as indicated by the poorer t-statistic. The unbiased Phase II estimates of the target mean, and the actually occurring squared residuals, suggested a good estimate of the 95% prediction intervals was given in this case also.

This is confirmed by inspection of Fig. 10 where the estimated prediction intervals (the blue dashed lines) correspond well with intervals calculated using the actually occurring residual errors (the green dashed lines). The option price series illustrated in Fig. 10 is

new unseen data. The Example# 3 model gives an unbiased price prediction for the series, with a t-statistic of -0.27 in an independent t-test assuming unequal variances. By comparison, the BS prediction of the option price is also unbiased for this series with a t-statistic of -0.20. In a paired t-test of the model predictions, the two models show no significant difference and the t-statistic is -1.12. In a paired t-test of the model residuals, a statistic of 1.12 is obtained. These results suggest that both the Example#3 model and the BS formula provide good models of the DGP for this sample. The relatively good result for the BS formula given the small sample of 21 observations is surprising, as the BS formula usually gives biased results for large samples. The Example#3 NN model though, has only two inputs, moneyness, and time to maturity, compared to the five inputs of the BS formula however.

To facilitate equal comparison with the BS formula a further NN model, Example #4, was trained. Example #4 used the same training and test data used for Example#3. For Example#4 however, all five of the BS inputs were used. In addition, the network architecture was optimized, using sensitivity based pruning, to give 4 hidden layer nodes. Table 12 gives the results for Example#4.

Table 12 Estimates by Proposed Network (Example#4)

Phase	Layers	Input	Output	R^2	Adj.R^2	F_Statistic		t-statistic		t-test
	[nodes]	x = [S,X,t,r,iv]				F_{crit}(1 tail)	F_{calc}(0.05)	t_{crit}(2 tail)	t_{calc}(0.05)	[bias]
Comparison with Actual Target (O)										
I	5-4-1	x	$\mu_y^*(x)$	0.995	0.997	1.04	1.02	1.96	-0.61	Unbiased
Comparison with Actual Target (O) & Squared Residuals										
II	5-4-2	x	$\mu_y^*(x)$	0.992	0.995	1.04	1.02	1.96	-0.44	Unbiased
II	5-4-2	x	$\sigma^{*2}(x)$	0.294	0.071	1.04	258.32	1.96	4.81	Biased
III	5-4-2	x	$\mu_y^*(x)$	0.990	0.994	1.04	1.04	1.96	-0.46	Unbiased
III	5-4-2	x	$\sigma^{*2}(x)$	0.335	0.312	1.04	7.27	1.96	2.40	Biased

Table 12 shows the proposed network produces an unbiased estimate, $\mu_y^*(x)$, of the mean of the target O in all three training phases. The estimate $\sigma^{*2}(x)$ is biased for both PhaseII and III.

In Table 12 the estimate $\mu_y^*(x)$ is again unbiased for all three phases. The fit to the observed option prices O, is very good with Phase II and III R^2 and Adj. R^2 figures >0.99, which is higher than the corresponding figures for Example#3. In addition, the F-test figures for the estimate $\mu_y^*(x)$, indicate no significant difference in the variance compared with the target O. However, the phase II estimate $\sigma^{*2}(x)$, is now biased, as well as the Phase III estimate. Inspection of the means of the actual squared residuals, and their estimates given by $\sigma^{*2}(x)$, indicate an underestimate. This result suggests the proposed method underestimates the magnitude of squared residuals when the fit to the target O (or d) is very good. However, it is unlikely to be a problem in practice, as

the underestimate only occurs when $\mu_y^*(\boldsymbol{x})$ is an unbiased estimate of the target O, *and* the fit is better than R^2 >0.99, *and* calculated F-statistics are less than their critical values[71].

Fig. 11 shows the results of applying Example#4 to the same option price series illustrated in Fig. 10. In Fig 11 the estimated prediction intervals (the blue dashed lines), correspond well with intervals calculated using the actually occurring residual errors (the green dashed lines), for the region of moneyness >1.05 where the fit of $\mu_y^*(\boldsymbol{x})$ to O is in general poorer than the region where moneyness is <1.05. However, the overall fit for Example#4 is much better than the fit for Example#3, which is itself unbiased. It can be seen that the BS price predictions are outside the prediction bands for the Example#4 predictions, for moneyness >1.0. Figures 10 and 11 graphically illustrate the performance of the proposed method for estimating prediction intervals, and the utility of prediction intervals for assessing the differences in performance over the input space of option pricing models.

[71] The method is based on the use of a least squares cost function. However, maximum likelihood and least squares estimators are equivalent in terms of performance, and the tendency for maximum likelihood estimators to underestimate the variance is known, and has been remarked in Bishop (1995) Chapter 6, section 6.3.

Fig. 10 Prediction Intervals for Example#3

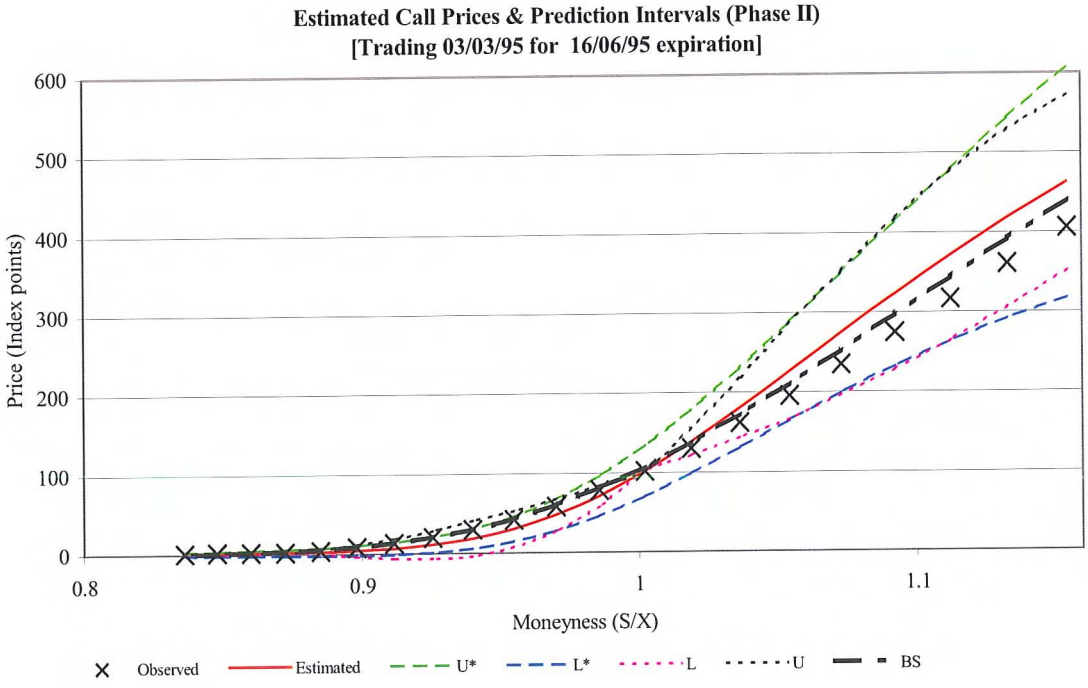
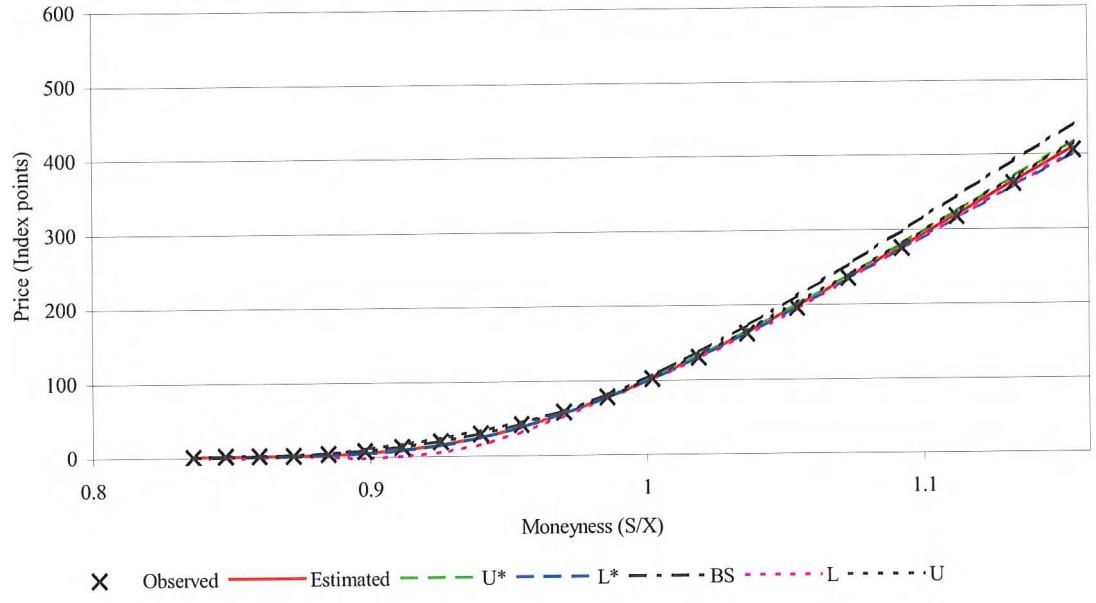


Fig. 11 Prediction Intervals for Example#4



Figs. 10 and 11. Prices of FT-SE100 Call Options trading on the 3rd March 1995 for June 1995 expiration. The crosses are the observed option prices. The red lines are the network estimates of the prices. The green and blue dashed lines are estimated upper and lower prediction intervals. The black and purple dashed lines are corresponding intervals calculated using the actually occurring residual errors. The heavy black dashed and dotted line is the BS price prediction.

3.6 Conclusion

In this chapter, a method for computing standard errors, confidence intervals, and prediction intervals, for any sufficiently flexible technique for non-linear regression is presented. The method rests on the classical framework for least squares regression and maximum likelihood estimation. The implementation of the theory described here is new, and relies on a special training algorithm. It can be applied to a broad class of computational knowledge discovery techniques for non-parametric non-linear regression, but was demonstrated here using neural nets.

The method was applied successfully to a standard synthetic set of data and gave statistically acceptable results. It performed comparably with a similar though non-general method proposed by Nix and Weigend (1995). A synthetic option pricing test was constructed and the true noise variance function recovered. In a test with actual option prices, an unbiased estimate of the actual squared errors for the fitted option prices was obtained. However, further tests with actual option prices suggested the noise variance is underestimated when Adj. R^2 is greater than 0.99, but this is a feature of all methods based on least squares or maximum likelihood. The theory presented, and the overall results of the tests, suggest that the method is appropriate for determining prediction intervals for target data with heteroskedastic errors.

Other possible applications within the options market domain include; Recovery of risk neutral densities from option prices, where the method may be useful in avoiding the limitations of bootstrap and Monte Carlo methods for computing statistical intervals[72]; Recovering the time dependent conditional densities of asset price series; And, volatility modelling. Possible extensions of the technique include the addition of extra output nodes to model higher moments, and thus recover a more complete description of the error distribution.

[72] Melick, W.R., and Thomas, C.P. 1998. "*Confidence Intervals and Constant Maturity Series for Probability Measures Extracted from Option Prices*". Paper presented at the conference 'Information Contained in Prices of Financial Assets', Bank of Canada.
Andersson, M., and Lomakka, M. 2003. *"Evaluating Implied RND's by Some New Confidence Interval Estimation Techniques"*, Sveriges Riksbank Working Paper Series, No. 146.

CHAPTER 4

Application of the Computational Framework

4.0 Introduction

In this chapter, demonstrations are given of key aspects of the Computational Framework and proposed methods for establishing statistical confidence in models and predictions, described in Chapters 2 and 3. This is done by practical application to questions of interest to options market practitioners and researchers. First, use of the GeTS Model Search Algorithm for Input Space Search (Table 3) is demonstrated in a case study concerning the question of option pricing under transaction costs. Then, selection of an appropriate model architecture is demonstrated using the Architecture Selection Algorithm (Table 4), and the Prediction Risk criterion discussed in Chapter 3 section 3.4.3. Next, use of computational knowledge discovery techniques as a tool to perform statistical tests, is demonstrated in a further investigation of transaction costs. Finally, the framework and methods are applied to the extraction of patterns implicit in options market data, which provide probabilistic information on *future* prices of the underlying assets. Namely, Risk Neutral Densities (RNDs). The object of these demonstrations, is to show that when computational knowledge discovery techniques are applied to options market data, within a systematic KDD framework incorporating statistically principled practical procedures, the result is improved statistical confidence in the models and results obtained, and fresh insights for the application domain.

4.1 Application I: Option Pricing

Three decades ago, the development of the Black-Scholes option pricing model, revolutionised options markets. Black and Scholes had the insight that prices of options could be modelled by a particular stochastic differential equation, now known as the BS differential equation. The famous BS formula is an exact solution to this differential equation. The BS formula applies only to European exercise options on non-dividend-paying assets and rests on a severely restrictive set of assumptions. There have been many attempts to extend and improve on the BS model, and a large academic literature exists. Extensions and improvements to the BS model documented in this literature are mostly based on alternative solutions to the BS differential equation. The majority of the extended and improved option-pricing models rely on analytical approximations or numerical methods, to solve the differential equation where no closed-form solution exists. However, as discussed below, the practical success of these models is questionable. The BS model retains its importance as a key benchmark, and the search for improved option pricing models remains a central concern of options market

researchers and practitioners. A review of the literature on option pricing is beyond the scope of this thesis, but can be found in Healy (2000)[73]. An explanation of the benchmark BS model is given in Appendix D. Hull (2000) provides a good description of the most important option pricing models. Here, comment is confined to empirical evidence on performance, and is presented in the next section.

4.1.1 Modern Parametric Option Pricing

The BS formula was the first successful tool for rational valuation of options. It was the first option pricing model where all the parameters were measurable. However, it and its extensions show systematic and substantial bias[74]. To improve pricing performance, the BS formula has been generalised to a class of models referred to here as *modern parametric option pricing models.* The developers of these modern parametric option pricing models hoped to obtain well-specified models, consistent with the dynamics of the underlying assets, and which were straightforward to estimate and consistently outperformed rival specifications. Arguably, their efforts were largely unsuccessful[75]. Even the most complex modern parametric models are outperformed by less general simpler models. They often produce parameters inconsistent with the time series of the underlying asset, give inferior hedging performance, and display systematic pricing biases[76]. To overcome these problems, there has been increasing interest among researchers in non-parametric techniques[77] and in model-free methods designed to discover/induce a model from data, including several of the computational knowledge discovery techniques. A review of the literature documenting applications of computational knowledge discovery techniques to option pricing follows in the next section.

---

[73] Healy, J. V., 2000. *" Data Mining, Machine Learning, and KDD: Applications to Option Market Data"*, discussion paper, Department of Computing Communication Systems and Mathematics, London Guildhall University.

[74] MacBeth, J., and Merville, L. 1979. *"An Empirical Examination of The Black-Scholes Call Option Pricing Model"*, Journal of Finance 34, pp 1173-1186.
Galai, D. 1983. "A survey of empirical tests of option-pricing models," in *Option Pricing: Theory and Applications* (M. Brenner, ed.), Lexington Books, Lexington, pp45-81.
Rubinstein, M. 1985. *"Nonparametric tests of alternative option pricing models using all reported trades and quotes on the 30 most active CBOE option classes from August 23, 1976 through August 31 1978"*, Journal of Finance 40, No. 2, pp 455-480.

[75] Lajbcygier P. 1999. *"Literature Review: The problem with modern Parametric Option Pricing"* Journal of Computational Intelligence in Finance Vol.7 No.5 pp6-23.

[76] Bakshi, G., Cao, C., and Chen, Z. 1997. *"Empirical performance of alternative option-pricing models"*, The Journal of Finance, Vol.52, No.5, pp2003-2049.
Bakshi, G., Cao, C., and Chen, Z. 1998. *"Pricing and hedging long-term options"*, Journal of Econometrics, pp 277-318.

[77] Pagan, A., Ullah, A. 1999. *Nonparametric Econometrics*, Cambridge University Press.

4.1.2 Literature Review: Computational Knowledge Discovery Techniques and Option Pricing

A number of researchers have applied computational knowledge discovery techniques to option pricing. Although these techniques were originally developed for entirely different purposes, they are well suited to this application. The mapping between option prices and their determinants is highly non-linear, and the price dynamics of the underlying assets on which option prices depend are not easily modelled to sufficient levels of accuracy. In addition, an important input to option pricing models, the asset price volatility is not directly observable. Computational knowledge discovery techniques provide an arguably superior alternative to modern parametric option pricing methods for dealing with these difficulties. They allow prior assumptions regarding distributional parameters to be avoided, making it possible to discard generalised diffusion processes entirely.

An early and influential application of computational knowledge discovery techniques to option market data is that of Hutchinson, Lo and Poggio (1994), who used three different methods. Namely, projection pursuit regression (PPR), radial basis functions (RBF), and multi layer perceptrons (MLP). The authors used a training set of 2 years of daily option prices generated by the BS formula, where the underlying stock price was simulated using the Monte Carlo method. They used a two dimensional model with moneyness, and time to maturity as inputs (S/X, t). They showed that training sets of as little as six months of daily data is sufficient to accurately recover the BS price, and to delta-hedge options.

The authors then applied the techniques to 5 years of daily closing prices for the Standard and Poor 500 index option to recover the implied pricing model and delta-hedge the options. They found "some evidence" that the computational knowledge discovery techniques outperformed the BS formula on this data. Paired t-tests were used for comparative tests of the delta-hedging performance of the different techniques and the BS formula. However, this was not done in the context of a systematic model search methodology, and the results were not adjusted for true significance. When the input space was partitioned into three parts, it was found that the three techniques performed differently in the different regions. Overall, the tests performed did not allow a conclusion as to which computational knowledge discovery technique was superior. Hutchinson, Lo, and Poggio did not attempt to optimise the architecture of the

methods they used. Four non-linear functions were used for each method, that is, four projections, basis functions, and hidden units respectively, for the PPR, RBF, and MLP. This number was chosen because of results obtained in preliminary trials. In their conclusion, the authors remark on the need for statistical techniques for network optimisation, the desirability of a data-dependent metric for model performance such as model prediction error, and the limitations of performance measures such as R^2. These are precisely the issues addressed in this thesis.

Kelly (1994)[78] used neural networks to price and hedge American style put options traded on four different common stocks. He used a data set of 1369 daily observations covering October 1 1993 to April 4 1994, and the closing daily bid price was used as the target. The resulting price estimates were compared with prices yielded by the Binomial Tree model of Cox, Ross, and Rubinstein (1979)[79]. Kelly found that neural networks valued and hedged American put options significantly better than the lattice based model. The neural nets had an R^2 of 0.996 and mean absolute error of $0.12 compared to 0.72 and $0.99 respectively for the Binomial Tree model.

Following the approach of Hutchinson, Lo and Poggio (1994), Herrmann and Narr (1997)[80] applied neural networks to the Dax index option traded on the Deutsche Terminborse (DTB), one of the world's most liquid index options. The Dax index option is a European exercise option and the Dax index is dividend adjusted with no dividends to consider. Thus, it is ideally suited for use of the BS model. Unlike Hutchinson, Lo and Poggio (1994) and Kelly (1994) Herrmann and Narr used synchronous time stamped transaction data (i.e. tick data) for the whole of 1995. The dataset used was provided by the Karlsruher Kapitalmarktdatenbank and contained more than 500,000 observations (records). Herrmann and Narr used their learning network to recover the market-implied pricing model from the dataset, and then compared its performance to the simple BS model. They concluded that neural nets are significantly better in explaining market prices than the BS formula, and that there were no observable signs of overfitting even when using networks with 11 hidden nodes.

---

[78] Kelly, D. 1994. "*Valuing and Hedging American Options Using Neural Networks*", Working paper, Carnegie Mellon University.

[79] See Appendix D for a description of the CRR model.

[80] Herrmann, R., and Narr, A. 1997. "*Neural Networks and the Valuation of Derivatives - Some Insights into the implied Pricing Mechanism of German Stock Index Options.*" Working paper, University of Karlsruhe, Institute for Decision Theory and Management Science, Department of Finance and Banking.

They also found that the fit of the networks in out-of-sample tests was sometimes even slightly better than the results for the training set. Since neural nets are twice differentiable, Herrmann and Narr were able to extract the implied RNDs. They found that the implied RNDs differed markedly from those obtained using the BS formula. They concluded that neural networks can discover the implied pricing mechanism of derivatives and are able to explain market prices far better than the BS formula. They characterised learning networks as a fast and easily handled valuation tool, once trained on market data, observing that additional input factors are easily incorporated.

Galindo (1999) followed up one suggestion for further research made by Hutchinson, Lo and Poggio (1994), that the relation between sample size and approximation error should be explored. He performed a comparative analysis of a variety of computational knowledge discovery techniques, the aim of which was to define a metric for comparison of different approaches. The methodology he developed is based on the study of error curves of the root mean square error (RMSE), as sample sizes and degrees of freedom, (e.g. numbers of hidden nodes, basis functions, projections etc) are varied. The methodology was tested by analysing the performance of a number of state-of-the-art techniques in recovering the BS formula from simulated option price data with added noise. Galindo found neural networks (MLPs) provided the best estimation with average RMSE of 0.7825 for a training sample of 6,000 records. OLS (used as a benchmark) was second best with average RMSE of 0.7861, and best for small samples with fewer than 1,125 records. K-nearest neighbour (KNN), and decision tree (CART), methods were worst with average RMSE of 0.8380 and 0.8721 respectively. When the BS assumptions were relaxed and interests rates and volatility were allowed to vary, Galindo found that the above performance ranking no longer held. Neural nets (MLPs) remained the best performer, but CART was now second, OLS with added explanatory variables third, standard OLS fourth, and KNN the worst performer. In summary, this study presented a framework for ranking performance, as the model capacity and training set size were varied for different computational knowledge discovery techniques. The performance rankings for the different techniques, displayed some sensitivity to the function (model) implicit in the data, however MLPs were found to be the best performer.

There have been a number of applications of genetic programming techniques to option pricing. These techniques are based on the random generation of a multiplicity of models. The 'fittest' (i.e. those with the best fit) are paired off using a combination algorithm, to form the next 'generation' of models. The process is repeated until a terminal model is obtained. The terminal model is considered the optimal solution for the particular modelling task. Genetic algorithms can be used with neural nets to create genetically adaptive neural nets (GANNs). GANNs use a form of bootstrap and have a low demand for data. It is claimed they overcome the problem of NNs finding local minima of the cost function, and always find the global minimum. GANNs are effectively a type of automated model search. They share with sensitivity based pruning the limitation that the true statistical significance of the terminal model is uncertain, since there is no way of knowing whether all models in the search path are statistically significant or properly nested.

White, Hatfield, and Dorsey (1998)[81] used GANNs to price options with futures-style margining, as traded on LIFFE. They found the pricing errors obtained were not significantly different from zero in t-tests, and the method was free from the pricing biases exhibited by current option pricing models for this class of option. Keber (1999)[82] applied genetic programming algorithms to the valuation of American put options on non-dividend paying stocks. His study used simulated data, and he found that his model outperformed other pricing methods described in the option pricing literature. Another application of genetic programming algorithms to option pricing is that of Chidambaran Jevons-Lee and Trigueros (1999)[83] who used Monte Carlo simulation to create a set of synthetic stock prices following a jump-diffusion process. They found that the genetic programming model approximated the true pricing formula better than the BS model. They also found that the genetic programming model outperformed a number of other models in a variety of settings. Lajbcygier Connor and

[81] White, A.J., Hatfield, G.B., and Dorsey, R.E. 1998. *"A Genetic Algorithm Approach to Pricing Options With Futures-style Margining"*, working paper, Murray State University, MS KY 42071.
[82] Keber, C. 1999. *"Option Valuation with the Genetic Programming Approach."* Proceedings of the 6th Conference on Computational Finance (formerly Neural Networks in the Capital Markets), Leonard N. Stern School of Business, New York University January 6-8, 1999.
[83] Chidambaran, K.N., Jevons-Lee, C.H., and Trigueros, J.R. 1999. *"An Adaptive Evolutionary Approach to Option Pricing via Genetic Programming"*, Proceedings of the 6th Conference on Computational Finance (formerly Neural Networks in the Capital Markets), Leonard N. Stern School of Business, New York University, January 6-8, 1999.

Tsang (1999)[84] proposed a new neural network architecture for option pricing which they called a 'constrained hybrid' approach. They identify a limitation of neural networks and other non-parametric option pricing methods. Namely, they do not use boundary condition information inherent in the data. The authors claim their method overcomes this limitation and can guarantee zero bias at the boundaries of the input space, thereby providing superior pricing performance to both conventional option pricing models and previously used neural network approaches.

Bennell and Sutcliffe (2000)[85] applied MLPs to LIFFE FTSE-100 index options. These options are ideally specified for application of the BS model. Daily closing prices were used. They concluded that NNs are superior in pricing performance to the BS model for options with a moneyness (S/X) less than 1.15, and maturity less than 200 days. Bennell and Sutcliffe studied the distribution of residual errors across the input space. They did not compute formal statistical intervals, or perform hypothesis tests for differences among the various models they considered, however.

Amilon (2000)[86] compared the BS model and MLPs in pricing and hedging Swedish stock index call options for the period 1997-1999. Daily data was used and models using both historical and implied volatility were tested. Both the bid and ask prices were used in order to model the spread (transaction costs) on the option premium rather than the more usual procedure of using the average of the two as the market price. MLPs using implied volatility were found to fit market prices better than MLPs using historical volatility, or the BS model using either implied or historical volatility. However, MLPs using historical volatility were found to give the best hedging performance. A moving block bootstrap procedure was used to compute confidence intervals, which were used to test the statistical significance of the results. Although the MLPs were found to produce superior fits to the data, the differences were often insignificant at the 5% level. This study is the only one reviewed here in which confidence intervals were computed and used for significance tests.

---

[84] Lajbcygier, P., Connor, J., and Tsang, R. 1999. *"A Constrained Hybrid Approach to Option Pricing"*, Proceedings of the 6th Conference on Computational Finance (formerly Neural Networks in the Capital Markets), Leonard N. Stern School of Business, New York University, January 6-8, 1999.

[85] Bennell, J. Sutcliffe, C. 2000. *"Black-Scholes Versus Artificial Neural Networks in Pricing FTSE 100 Options"* discussion paper 00-156, School of Management, Southampton University.

[86] Amilon H. 2001. *"A Neural Network Versus Black-Scholes: A Comparison of Pricing and Hedging Performances"* Working Paper Department of Economics, Lund University, Lund, Sweden.

With certain exceptions, the claims for superior option pricing performance of MLPs and other computational knowledge discovery techniques in the works reviewed above were based on ranking of competing models by global R^2 and error measures. The rankings were usually based on results for a single sample. Ensemble techniques and resampling (i.e. cross-validation, bootstrapping) were rarely used. Moreover, hypothesis tests for significant differences between models were generally not performed. Exceptions are: Amilon (2000), who computed statistical intervals using a bootstrap procedure, and used the resulting confidence intervals as the basis of tests for significance. White, Hatfield, and Dorsey (1998) whose results are based on t-tests. Hutchinson, Lo and Poggio (1994), who performed a small number of simple t-tests. And Galindo (1999), who used bootstrap replicates to produce averaged values of the RMSE for his work. An overall KDD framework, systematic data mining model search methodology, and true as opposed to nominal statistical significance levels, were not used in any of these studies.

4.2 Pricing FTSE 100 Index Options

The demonstrations in this section are performed using data relating to options on the FTSE 100 Index. The FTSE 100 Index is a share index of the 100 largest listed UK companies. The value of the index is a weighted average of the market capitalisation (i.e. number of shares × share value) of each company. The index is updated at one-minute intervals between 0830 and 1630 daily. It is a broad index, which accounts for 75%, by capitalisation, of the FT-Actuaries All Share Index. The FTSE 100 Index exhibits a 99.1% correlation with the FT-Actuaries All Share Index, which accounts for its widespread acceptance as a proxy for the whole market. Options on the FTSE 100 Index are chosen for study because they are effectively options on the market as a whole. They are highly liquid instruments for which data is readily available. Because of their payoff characteristics, quoted prices of FTSE 100 index options can be interpreted as realisations of the market's information and expectations concerning future price levels of the index. Moreover, these options have an important role in risk management, and are frequently used to hedge index tracking portfolios.

4.2.1 The Data: Selection, Cleaning and other Preparation Phases

Brief details of the data selection, transformations to the data, and nature of the data sets output from each of the data preparation phases prior to the data mining phase, are noted in this section.

Data Selection Phase

Output: Project Database

The option market data used was obtained from LIFFE's Market Data Service. The data set consists of daily closing prices for FTSE-100 index options for all trading dates from 13 March 1992 to 1 April 1997. The raw data set contained 119,413 European exercise (ESX) calls, and the same number of puts. A corresponding set of 153,710 American exercise (SEI) options was included. A small sample of this data is shown in Table 13.

Data Cleaning Phase

Output: Cleaned Data Set

The data was cleaned to exclude options with invalid or missing parameters. Only options that had actually been traded, as indicated by positive values of bid-ask spread, trading volume, and open interest (number of unclosed transactions) were used. To exclude outliers, options with moneyness (S/X) below 0.8 and above 1.4 were omitted. The volatility measure used was LIFFE tabulated at-the-money implied volatility (IV). There is evidence[87] that this is a good predictor of actual future volatility, and it gives a better fit as measured by Adj. R^2 than either historic volatility or alternative IV measures. Options with at-the-money implied volatilities (as tabulated by LIFFE) greater than 40% were excluded. The monthly Exchange Delivery Settlement Price (EDSP) obtained from LIFFE was included in the data set.

Data Reduction / Enrichment Phase

Output: Enriched Dataset

All quoted London interbank offer rates (LIBOR) available from Datastream were added to the dataset. The 3-month rate was selected as a proxy for the risk-free interest rate, as it matched the average maturity of the options. In addition, the FTSE 100 closing Price, and dividend yield, from Datastream were included.

---

[87] Healy, J. V. 1999. *"Volatility Implied in FTSE 100 Index Option Price Series As a Future Volatility Predictor: A New Approach"*, MSc Dissertation, Department of Economics and Finance, Brunel University London.

Table 13 Small Sample of LIFFE FTSE 100 Index Option Raw Data

Index	Trading_Date	Physical_Commodity	Expiry	Contract_Type	Exercise_Price	Volume	Open_Interest	Open_Price	Open_Range_First	Open_Range_Last	Daily_High	Daily_Low	Settlement_Price	Instrument_Settlement_Price	Volatility	ATM_Volatility	Closing_Bid	Closing_Offer
4776	08/14/92	ESX	09/01/92	P	2375	2	0				55	55	55	2375	19.18	19.18	53	58
6148	07/01/92	SEI	10/01/92	C	2600	3	3				55	47	44	2490	13.55	14.02	42	47
6148	07/01/92	SEI	10/01/92	P	2650	0	0						185	2490	18.87	16.13	183	188
12296	12/24/92	SEI	01/01/93	C	2900	106	5096				28	18	21	2827	18.72	18.99	19	24
12296	12/24/92	SEI	01/01/93	P	2950	0	0						124	2827	14.46	16.03	122	127
14328	03/11/93	ESX	05/01/93	C	2625	0	200						338	2962	17.31	14.72	336	341
18444	11/30/92	SEI	06/01/93	C	2400	0	293						440	2778	22.14	17.06	435	445
18444	11/30/92	SEI	06/01/93	P	2500	0	868						36	2778	18.24	15.81	30	42
19104	05/25/93	ESX	08/01/93	C	2525	0	0						338	2859	18.28	13.83	335	342
19104	05/25/93	ESX	08/01/93	P	2525	0	0						7	2859	17.02	13.74	5	9
23880	10/12/93	ESX	10/01/93	C	2725	0	0						368	3093	52.28	10.87	366	371
23880	10/12/93	ESX	10/01/93	P	2725	0	0						0	3093	49.47	9.35	0	1
24592	09/02/93	SEI	09/01/93	C	3100	1286	4473				23	20	21	3072	11.83	12.82	21	22
28656	12/20/93	ESX	02/01/94	C	3425	0	336						54	3393	12.63	12.87	51	57
28656	12/20/93	ESX	02/01/94	P	3425	0	4						85	3393	12.59	12.78	83	88
30740	11/25/93	SEI	12/01/93	P	3250	0	540						159	3092	0.5	15.01	155	160
33432	03/03/94	ESX	05/01/94	P	3325	0	100						150	3241	17.46	18.12	147	154
36888	03/24/94	SEI	04/01/94	C	2900	0	0						228	3124	23.05	19.34	225	232
38208	05/16/94	ESX	07/01/94	C	3325	0	0						17	3113	15.74	16.35	14	20
38208	05/16/94	ESX	07/01/94	P	3325	0	0						227	3113	15.26	16.37	224	231
42984	07/27/94	ESX	09/01/94	C	3825	0	372	0	0	0	0	0	1	3087	24.16	16.22	0	2
43036	04/06/94	SEI	07/01/94	C	3050	0	1						163	3132	16.7	16.89	158	169
43036	04/06/94	SEI	07/01/94	P	3100	2	14				98	96	100	3132	18.59	17.88	95	105
47760	08/02/94	ESX	12/01/94	C	3425	0	4195	0	0	0	0	0	44	3195	15.67	16.32	34	54
49184	10/03/94	SEI	10/01/94	C	3300	3	3374	0	0	0	1.5	1	1	2984	22.14	19.19	0	1
49184	10/03/94	SEI	10/01/94	P	3350	0	0	0	0	0	0	0	374	2984	0.3	19	371	377
52536	04/28/94	ESX	03/01/95	C	2925	0	0						335	3167	16.57	15.9	322	348
52536	04/28/94	ESX	03/01/95	P	2925	0	0						104	3167	17.65	15.91	93	115
55332	10/31/94	SEI	01/01/95	C	3150	12	5908	0	0	0	98	90	89	3097	17.88	18.47	86	93
55332	10/31/94	SEI	01/01/95	P	3200	0	5	0	0	0	0	0	152	3097	17.34	18.59	148	156
57312	03/03/95	ESX	05/01/95	C	3225	0	45	0	0	0	0	0	17	3030	13.93	14.68	14	21
57312	03/03/95	ESX	05/01/95	P	3225	0	5	0	0	0	0	0	212	3030	13.93	14.68	209	216
61480	03/15/95	SEI	04/01/95	C	2650	0	0	0	0	0	0	0	398	3048	31.78	15.04	395	401
61480	03/15/95	SEI	04/01/95	P	2700	0	1217	0	0	0	0	0	1	3048	18.98	15.14	0	3
62088	04/26/95	ESX	07/01/95	C	3275	0	1	0	0	0	0	0	68	3243	13.41	13.53	65	72
62088	04/26/95	ESX	07/01/95	P	3275	0	0	0	0	0	0	0	100	3243	13.41	13.5	96	104
66864	06/22/95	ESX	09/01/95	P	3175	0	500	0	0	0	0	0	12	3431	13.62	12.29	10	15
67628	04/04/95	SEI	07/01/95	C	3200	0	1200	0	0	0	0	0	115	3186	15.27	15.31	110	120
73776	09/01/95	SEI	10/01/95	P	3550	25	229	0	0	0	90	79	76	3510	11.22	11.47	73	79
76416	12/07/95	ESX	01/01/96	C	3675	380	1363	0	0	0	59	50	47	3652	11.54	11.66	44	50
76416	12/07/95	ESX	01/01/96	P	3675	380	1447	0	0	0	67	62	69	3652	11.58	11.71	67	72
79924	10/11/95	SEI	12/01/95	C	3700	51	5446	0	0	0	16	14	16	3471	13.64	15.11	13	19
81192	12/28/95	ESX	03/01/96	C	2675	0	0	0	0	0	0	0	1002	3691	19.77	11.63	998	1006
81192	12/28/95	ESX	03/01/96	P	2675	0	0	0	0	0	0	0	1	3691	30.06	11.61	0	2
85968	07/14/95	ESX	06/01/96	C	3525	0	10	0	0	0	0	0	172	3504	14.35	14.43	160	185
86072	02/27/96	SEI	03/01/96	C	3350	0	0	0	0	0	0	0	369	3716	17.2	12.26	0	0
86072	02/27/96	SEI	03/01/96	P	3400	0	3753	0	0	0	0	0	1	3716	19.93	11.93	0	0
90744	05/21/96	ESX	07/01/96	C	3325	0	0	0	0	0	0	0	463	3792	13.56	11.4	0	0
90744	05/21/96	ESX	07/01/96	P	3325	0	0	0	0	0	0	0	1	3792	15.68	11.37	0	0
92220	02/15/96	SEI	06/01/96	C	2900	0	0	0	0	0	0	0	877	3777	25.74	12.24	0	0
92220	02/15/96	SEI	06/01/96	P	3000	0	752	0	0	0	0	0	3	3777	19.86	12.2	0	0
95520	07/03/96	ESX	09/01/96	C	3475	0	10	0	0	0	0	0	242	3704	13.1	11.48	0	0
95520	07/03/96	ESX	09/01/96	P	3475	0	987	0	0	0	0	0	16	3704	13.12	11.52	0	0
98368	08/12/96	SEI	08/01/96	C	3650	0	401						153	3803	21.61	11.16	150	155
98368	08/12/96	SEI	08/01/96	P	3700	101	5258				1.5	1	1	3803	16.26	11.39	1	1.5
100296	02/21/96	ESX	12/01/96	C	3925	0	1056	0	0	0	0	0	106	3765	12.92	13.51	0	0
100296	02/21/96	ESX	12/01/96	P	3925	0	0	0	0	0	0	0	258	3765	12.92	13.51	0	0
109848	12/31/96	ESX	03/01/97	C	3425	0	8						692	4124	19.06	11.91		
109848	12/31/96	ESX	03/01/97	P	3425	0	3290						3	4124	19.91	11.92		
116812	04/18/97	SEI	04/01/97	C	3300	0	0						995	4306	0	0		
119400	03/27/97	ESX	03/01/98	C	4825	0	0						83	4413	12.84	13.61	70	80
119400	03/27/97	ESX	03/01/98	P	4825	0	0						495	4413	12.84	13.61		
122960	05/16/97	SEI	06/01/97	P	3850	0	110						1	4690	27.5	12.48		
129108	06/09/97	SEI	09/01/97	C	4750	0	4						135	4684	14.24	14.53		
129108	06/09/97	SEI	09/01/97	P	4800	0	0						180	4684	14.03	14.54		
141404	09/19/97	SEI	12/01/97	C	5350	0	0						92	5029	18.62	20.87		
141404	09/19/97	SEI	12/01/97	P	5400	0	2						402	5029	18.35	20.49		
147552	12/24/97	SEI	01/01/98	C	4100	0	0						931	5026	62.46	25.24		
147552	12/24/97	SEI	01/01/98	P	4150	0	4						10	5026	44.33	24.02		
153700	12/31/97	SEI	12/01/98	C	4900	0	0						741	5135	25.68	25.27		

Data Preparation Phase

Outputs:

Derived Variables

The following derived variables were generated in the data set; Moneyness (S/X), and the FTSE 100 closing price on the expiration date of the option (Terminal FTSE 100). In addition, as bid, ask, and spread, for the FTSE 100 index are not quoted, they were computed. This was done by using individual bid and ask prices, and numbers of shares in issue for the constituent stocks comprising the index, together with data on mergers and acquisitions, stock splits, new issues, and listings and delistings. This data was obtained from Datastream.

Training Sets and Test Sets

The data was randomly partitioned into training sets and test sets appropriate for each of the data mining exercises to be undertaken. A special test set of constant maturity, non-overlapping, complete option price series, for each available expiration month was also constructed for use in connection with risk neutral densities. This is discussed further in section 4.4.2.

4.2.2 Data Mining Model Search: Option Pricing with Transaction Costs[88]

One of the most unrealistic simplifying assumptions underlying the majority of modern parametric option pricing models, is that markets are 'frictionless'. That is, there are no transaction costs, taxes, or other factors that affect market prices. In reality, market frictions exist. However, their effect on option prices is an open question. This question was investigated, and the results are presented here as a case study, to demonstrate a data mining model search performed using the GeTS Model Search Algorithm for Input Space Search described in Chapter 2, section 2.4.5 Table 3.

Step 1. A general unrestricted model (GUM) was formulated. In this case, the GUM contained the five inputs to the BS formula plus three extra inputs. The three extra inputs were: 1) Spread, the difference between the bid and offer price for the option. 2) Volume, the number of contracts traded for the option. 3) Open interest (OI), the number of contracts traded but not yet 'closed out' for the option. Spread was included

[88] An earlier version of the material in this section appeared in; Healy, J.V., Dixon, M., Read, B.J., and Cai, F.F. 2001. *"A Data-Centric Approach to Understanding the Pricing of Financial Options"*, European Physics Journal B: Proceedings of the 3rd International Conference on Applications of Physics to Financial Analysis 2001.

because it is a transaction cost which is paid by the purchaser of the option. Volume and OI were included as measures of liquidity of a particular option. The GUM thus takes the form.

$$C = f(S, X, t, r, IV, Spread, Volume, OI) \tag{4.1}$$

In equation (4.1), C,S,X,t,r, and IV, are the five standard BS inputs as defined in Appendix D. The object of the model search was to determine whether, and to what degree, the variables *Spread*, *Volume* and *OI* affect option prices. A model not including these variables, assumes the price C is unaffected by the liquidity of the option, or the transaction costs involved in its purchase.

Step 2. The input variables were fitted to the target (response variable) C for the GUM, using OLS regression. Table 14 shows results and diagnostic statistics.

Table 14 Step 2: Results for OLS regression of GUM

Response Variable: C		**Sample: 7212**		**Method: OLS**
Variable	**Coefficient**	**Std. Error**	**t-Stat.**	**Prob.**
INTERCEPT	-55.46	7.22	-7.68	0.0000
S	0.61	0.00	214.28	0.0000
X	-0.59	0.00	-212.73	0.0000
t	219.54	3.18	69.04	0.0000
r	-284.43	42.24	-6.73	0.0000
IV	490.28	20.28	24.17	0.0000
SPREAD	-2.12	0.14	-14.93	0.0000
VOLUME	0.00	0.00	-3.23	0.0013
OI	0.00	0.00	-7.88	0.0000
R-squared	0.89	F-statistic		7150.76
Adjusted R-squared	0.89	Prob(F-statistic)		0.00
S.E. of regression	35.35	Durbin-Watson stat		1.74
Diagnostics				
Jarque-Bera Statistic	31216.94	Probability		0.00
Ramsey RESET test:				
F-statistic	4115.98	Probability		0.00
Log likelihood ratio	7203.26	Probability		0.00
White Heteroskedasticity Test:				
F-statistic	51.08	Probability		0.00
Obs*R-squared	735.60	Probability		0.00
Breusch-Godfrey Serial Correlation LM Test:				
F-statistic	86.21	Probability		0.00
Obs*R-squared	250.06	Probability		0.00

Table 14 Results of an OLS regression for the GUM defined by equation (4.1).

The t-statistic, which is computed as the ratio of an estimated coefficient to its standard error, is used to test the hypothesis that a coefficient is equal to zero. The last column of the output gives the p-values for the observed t-statistics. At the 5% significance level in a two-tailed test, a p-value lower than 0.05 is taken as evidence to reject the null hypothesis of a zero coefficient. Here, all the p-values were less than 0.05 suggesting that all the coefficients were different to zero. Hence, all the input variables were

significant. The F-test is a *joint* test of the null hypothesis that *all* the coefficients (excepting the intercept) are zero, even if all the t-statistics are insignificant, an F-statistic can still be highly significant. Here, the p-value for the F-statistic was less than the significance level of 0.05, and the null hypothesis was rejected. This result suggested that *at least one* of the coefficients was different to zero. The Durbin-Watson statistic measures the serial correlation in the residuals, a value of less than 2 is indicative of positive serial correlation. Jarque-Bera is a test statistic for testing whether a series is normally distributed. The small p-value for this statistic suggested the null hypothesis of normality of the residuals should be rejected. The small p-values for White's test and the LM test suggested the existence of heteroskedasticity, and serial correlation of the residuals, respectively. The RESET test is a test of functional form applicable to OLS, here the results suggested an incorrect functional form. These results support the use of appropriate non-linear techniques for the remainder of the model search.

Step 3. A MLP with a single hidden layer was trained using the GUM inputs. An initial topology of 4 hidden layer nodes was selected in accordance with equation (2.5). F and t tests were performed on the actual and fitted values of the target using the independent test set. The residuals were tested for normality using the Jarque-Bera test. Results are presented in Table 15.

Table 15 Step 3: Results for MLP fit of GUM specification

Model	Topology	R^2	Adj. R^2	F-stat.		t-stat.	t-test	Jarque-Bera stat.
[GUM]	[nodes]			F_{crit}(1 tail)	F_{calc}(0.05)	[t_{crit}(2,0.05) = 1.96]	[bias]	[crit. value = 5.99]
C(S,X,t,r,IV,spread,volume,OI)	8-4-1	0.996	0.998	1.04	1.01	-0.218	Unbiased	10256784.23

Table 15 Results of fitting the GUM specification given in equation (4.1) to call option prices.

The Jarque-Bera statistic confirmed the residuals were not normally distributed. This test was not repeated for further stages of the model search. The F-test result suggested there was no significant difference between the variances of the actual and fitted values of the target variable. A t-test, assuming no difference in the variances, indicated that the fitted values were unbiased estimates of the means of the actual target values, and there was no difference at the 95% confidence level. This model was designated as the CURRENT MODEL, and the model search proceeded to Step 4.

Step 4. Step 4 was repeated for 5 iterations. Variable deletion tests were performed to test restrictions on the GUM created by omitting the variables *spread*, *volume*, *OI*, *r*, and *IV* respectively. Tests were not performed for *t*, *S*, and *X*, as these variables were

already well known as important determinants of option price. The variables r, and IV, were included in the tests for comparison purposes. Table 16 presents the results of the variable deletion tests on the restricted models. The figures relate to comparisons of the residuals of each restricted model in turn with the GUM.

Table 16 Step 4: Variable Deletion Tests Based on Comparison of Residuals

Model	Topology	R^2	Adj. R^2	t-stat (paired).	Paired t-test	Delete?
[comparison with GUM]	[nodes]			$[t_{crit}(2,0.05) = 1.96]$	[H0:=no difference]	
GUM-volume	7-4-1	0.936	0.966	-8.25	Reject	No
GUM-r	7-4-1	0.906	0.952	-23.47	Reject	No
GUM-OI	7-4-1	0.882	0.938	-8.89	Reject	No
GUM-spread	7-4-1	0.827	0.909	-4.93	Reject	No
GUM-IV	7-4-1	0.563	0.640	-6.07	Reject	No

Table 16 Variable deletion tests of restrictions on the GUM. Results are ranked by Adj.R^2. The tests consist of a paired t-test of residuals of the GUM and the restricted model. The null hypothesis of no difference is tested. Acceptance of the null hypothesis implies a variable makes no significant difference to the model and can be omitted. Rejection suggests the variable is significant and should be retained if the model is to be correctly specified. F-statistics are not reported, as they require normality of the residuals. The Jarque-Bera test in Step 3 has indicated the residuals are not normally distributed.

The results in Table 16 suggest *all* the variables tested are statistically significant for option prices, since the null hypothesis of no difference in the residuals is rejected in each case. It follows that a correctly specified pricing model for the options considered should include the variables spread, volume, and OI. The BS formula, Black-I formula, and modern parametric option pricing models do not include these variables. Thus, they are mis-specified models for these options, which are amongst the most widely traded and liquid in the UK market. A variable that is not statistically significant does not influence a model and should be deleted. However, a variable may be statistically significant, and yet have little effect on the fit to the target (response variable). To determine the influence of each of the variables subjected to the variable deletion tests, on the fit to the target, further tests were performed. Results are presented in Table 17. The figures relate to the ability of the GUM and the restricted models to fit option prices.

Table 17 Explanatory Power of Restricted Models for Dependent Variable

Model	Topology	R^2	Adj. R^2	t-stat.	t-test	Omitted Var.
[comparison with target]	[nodes]			$[t_{crit}(2,0.05) = 1.96]$	[bias]	[Adj. R^2: GUM-RM.]
GUM	8-4-1	0.996	0.998	-0.22	Unbiased	-
GUM-r	7-4-1	0.996	0.998	0.09	Unbiased	0.00003
GUM-OI	7-4-1	0.996	0.998	-0.08	Unbiased	0.00026
GUM-spread	7-4-1	0.996	0.998	-0.12	Unbiased	0.00028
GUM-volume	7-4-1	0.996	0.998	-0.12	Unbiased	0.00029
GUM-IV	7-4-1	0.994	0.996	0.01	Unbiased	0.00168

Table 17 Fit to target (response variable) of GUM and restricted models. Results are ranked by Adj.R^2. For each model, t-statistics for fit to the target are reported. Row 1 reports results for the GUM, the remaining rows report results for the restricted models. The right hand column headed 'Omitted Var.' reports the amount of variance in the target variable 'explained' by each omitted variable. The amount of variance in the target unexplained by the GUM is 0.002 (to 3 d.p.).

Table 17 demonstrates that both the GUM and *all* the restricted models produced unbiased estimates of the mean of the target, at the 95% significance level. Virtually all the variance in the response variable was explained by the models. The Adj. R^2 was identical to 3 decimal places at 0.998 for all the models, excepting that which omitted *IV*. Only approximately 0.002 of the variance of the target was unaccounted for by the GUM. Interestingly, the results in the last column indicated that *r*, the risk-free interest rate, had around ten times *less* explanatory power for option prices than the variables *spread*, *volume*, and *OI*. The latter three are customarily omitted from modern parametric option pricing models. The variable *IV*, had the most explanatory power. The explanatory power of *all* of these variables for the target though, was very small. From the perspective of statistical rigour however, a model that omits any of these variables is a mis-specified model. Nevertheless, given they contribute so little to the fit, it may be expedient to omit one or all of these variables and estimate a mis-specified model. This is because the rate of convergence of non-parametric estimators increases considerably as the number of inputs decreases. In addition, the extra restrictions may help avoid overfitting in the learning algorithm, and improve out-of-sample performance.

Step 5 The variable deletion tests performed in Step 4, reported in Table 16, indicated that no variables should be deleted from the GUM given by equation (4.1). The GUM was therefore the single terminal model resulting from Step 4. Consequently, the GUM was accepted as the parsimonious undominated encompassing model.

Step 6 At this stage the model search may (optionally) proceed using the Architecture Selection Algorithm described in Chapter 2 section 2.4.5 Table 4. Practical use of the Architecture Selection Algorithm is described in Section 4.2.3.

Assuming the data mining model search stops at Step 5, the true significance level for the parsimonious undominated encompassing model, is now calculated in accordance with the procedures described in Chapter 2, section 2.4.5, and the true and nominal significance levels given in Table 5. In computing true significance levels, if *j* hypothesis tests are required to reach the final model, then the true significance of the j^{th} test in the sequence is given by equation (2.6). It is important to understand however, that only tests which result in the *acceptance* of a restricted model are counted. A test

that leads to the rejection of a restricted model is therefore not counted. In the case of the GUM given by equation (4.1) hypothesis test results led to rejection of the OLS version of the GUM, and all of the restricted MLP models. Only in the case of the MLP version of the GUM, did the hypothesis test lead to acceptance of the model. It follows that only *one* hypothesis test was required to reach the final model, so that the true and nominal significance levels are identical in this case.

If the data mining model search proceeds to Step 6 and the Architecture Selection Algorithm is implemented, the true significance level should be computed for the model resulting from that procedure. Since there will then be at least one extra hypothesis test required to reach the final model, the true and nominal significance levels will differ.

4.2.3 NN Option Pricing Models: Architecture Selection

In this section, use of the architecture selection algorithm given in Chapter 2, section 2.4.5, Table 4, is demonstrated. The goal of the algorithm is to choose from a set of models with competing architectures, the one giving the best generalisation performance. This algorithm is used together with estimates of the Prediction Risk (*PR*) obtained by *test set validation* using equation (3.31), to select the model having the smallest value of *PR*, from amongst those considered. A systematic search of a subset of the space of possible architectures for a MLP with a single hidden layer was performed using the algorithm. The training algorithm given in Chapter 3, section 3.5, Table 6, was then used together with equation (3.33), to construct a model which permits estimation of *PR* where target values are unknown, without using either test set validation, or cross validation.

The data used for the example which follows was the same set of daily price data for the FTSE 100 index 'ESX' European exercise call options, used for Example#2, 3, and 4 in Chapter 3. The available data was randomly partitioned into an equal sized training set and test set. The test set was designated set 'C'. The training set was then randomly partitioned into two further training sets designated as sets 'A' and 'B' respectively.

Set A was used as the training data for a set of 17 models. Only architectures with 4 to 20 hidden units were considered, as the model that parsimoniously minimises prediction risk was most likely to be in this range. The models differed only in numbers of hidden

units, the training dataset, input variables, and the seed controlling the set of random initial weights, were not varied. First, a model with 4 hidden layer nodes was trained. Thereafter, the number of hidden layer nodes was incremented by one for each model trained, up to the pre-selected maximum of 20. A sequence of 'nested' models was thus created. The *PR* for each model was estimated using equation (3.31) on the test set C (test set validation). The architecture giving the best generalisation (lowest *PR*) for the sequence could now be readily identified. In this case, the model having 15 hidden layer nodes gave the lowest value for *PR*.

The training algorithm given in Table 6 of Chapter 3 was now applied. The model with 15 hidden layer nodes was used as the Phase I model, and applied to training set B to generate a set of residuals for Phase II training. A set of 6 Phase II models with numbers of hidden layer nodes incrementally increasing from 15 to 20 were trained. The *PR* for each model was estimated using equation (3.33) and test set C. The *PR* values for the Phase II models suggested that negligible improvements in performance were obtainable by adding further hidden layer nodes in Phase II. Hence, the Phase II model with 15 hidden layer nodes was accepted as giving the best generalisation performance. Results of the architecture selection process using the algorithms given in Table 4 and Table 6 are presented in Table 18. When applied to previously unseen inputs, for which the corresponding target values are unknown, an estimate of *PR* is obtainable from the Phase II model using equation (3.33). Test set validation, or cross validation, are not required.

Table 18 Example of Architecture Selection using Prediction Risk

Hidden layer Nodes	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Phase I [Eqn. (3.31)]	52.8	47.0	47.2	40.8	44.4	56.0	46.3	40.2	47.1	39.6	35.6	**31.4**	41.3	42.6	46.5	38.4	40.7
Phase II [Eqn. (3.33)]	-	-	-	-	-	-	-	-	-	-	-	**9.9**	9.3	9.6	9.3	9.3	9.2

Table 18 shows how *PR* can be used to select the number of hidden layer nodes for a MLP with a single hidden layer. The architecture selection algorithm (Table 4) was applied and 17 models having between 4 and 20 hidden layer nodes were created. Equation (3.31) was used to estimate the *PR*, ($E[MSE]$) for each model. The model with 15 hidden layer nodes was identified as giving the best generalization performance (lowest *PR*). This was used as the Phase I model in the training algorithm given in Table 6, and a set of Phase II models was created. Equation (3.3) was used to estimate *PR* for each Phase II model. The results suggest there is no significant gain by adding further hidden layer nodes in Phase II.

It is important to stress here that the Phase II model using equation (3.33) allows prediction risk to be estimated for unknown targets, without use of cross validation or test set validation. No claims of superior performance compared to alternative criteria for architecture selection are intended. The reason for preferring *PR* to alternatives is because it is a statistical criterion which permits selection of a topology which

minimises the (expected) sum of squared errors on test data from a set of trained models with different topologies.

4.2.4 Hedging With Transaction Costs: NNs As an Investigative Tool

In Section 4.2.2 a data Mining Model Search algorithm was used, together with a GUM specification implemented in non-linear non-parametric form using a MLP, to show that a correctly specified option pricing model should include the spread on the option premium, volume of contracts traded, and open contracts (*OI*), as input variables. These variables, though statistically significant, were found to have little effect on option prices however. In this section, the MLP is used as an investigative tool to empirically examine the effects of including transaction costs for trading the underlying security in option pricing models. Ignoring these latter costs and using the mid price of the underlying asset for hedging (pricing) is incorrect in principle. It is an open question whether incorporating them will improve pricing and hedging performance in practice.

The BS and most other option pricing models assume there are no transaction costs involved in trading the underlying security[89]. However, certain authors have proposed models which proportionally increase the price of the underlying asset to reflect these transaction costs. Leland (1985)[90] considered hedging at fixed time intervals. His pricing model approximately hedges an option, and matches the payoff at maturity. A fixed time interval is also used in the binomial tree model of Boyle and Vorst (1992)[91]. Hoggard, Whalley and Wilmott (1994)[92] have extended the Leland fixed time interval approach to a more general cost structure. Henrotte (1993) and Toft (1996) have shown that models where hedge rebalancing occurs at variable intervals, triggered by a percentage change in the price of the underlying asset, are outperformed by fixed time interval models of the Leland type[93]. To obtain optimal hedging performance, Hodges and Neuberger (1989) and Davis, Panas and Zariphopoulou (1993) considered hedging

---

[89] See discussion in Appendix D.

[90] Leland, H. E. 1985. *'Option Pricing and Replication with Transaction Costs'*, The Journal of Finance 40, pp1283–1301.

[91] Boyle, P.P., and Vorst, T. 1992. *'Option Replication in Discrete Time with Transaction Costs'*, The Journal of Finance 47, pp271–293.

[92] Hoggard, T., Whalley, A.E., and Wilmott, P. 1994. *'Hedging Option Portfolios in the Presence of Transaction Costs'*, Advances in Futures and Options. 7, pp21-35.

[93] Henrotte, P. 1993. *'Transaction Costs and Duplication Strategies'*, working paper, Stanford University and Groupe HEC.

Toft, K. 1996. *'On the Mean-Variance Trade-Off in Option Replication with Transaction Costs'*, Journal of Financial and Quantitative Analysis 31, pp233–263.

in the presence of transaction costs as a stochastic optimal control problem[94]. The optimal control models allow identification of a bounded area within which no hedging should occur. Trading should only occur to bring the hedge ratio back to the nearest boundary when it lies outside this area. However, none of the above approaches to transaction costs reflects the market reality. This is because they assume the correct underlying asset price is the mid-price, and then increase this by a constant proportion. In real financial markets, the original data consists of tick-by-tick quotations of ask and bid prices. The mid-price is an artefact derived from the original ask and bid prices. In organised financial markets, the price an investor pays to *purchase* a security is the ask price. The price an investor receives for the *sale* of a security is the bid price. An exchange traded asset thus has two prices rather than a single unique price. The spread between the ask and the bid is the compensation received by the market makers for providing liquidity by standing ready to sell (purchase) securities. Ask and bid prices, and the spread, vary with time. The use of proportional transaction costs ignores the existence of these two separate prices, and fails to capture their variances.

A sufficiently liquid and frequently traded option was required, in order to test the significance of bid and ask prices of the underlying asset for option prices, and hence for hedging options positions. Thus, ESX European exercise options (both calls and puts) on the FTSE 100 index were chosen, rather than individual equity options. However the FTSE 100 index cannot itself be traded, so no ask and bid prices are quoted for it. It was first necessary to estimate ask and bid prices for the index. The FTSE-100 index is a value weighted index of the mid (average of the ask and bid) prices of its constituent stocks. The weighting factors are the market values (capitalisation) of the stocks concerned. The market value of each stock is the product of the stock price and the number of shares in issue. The daily closing price of the FTSE-100 index is available through Datastream. The (daily closing) ask price, bid price, and number of shares in issue, are also available through Datastream for the individual constituent stocks comprising the FTSE-100 index. Using this data, separate value weighted indices of the FTSE-100 ask and bid prices were constructed.

[94] Hodges, S. D., and Neuberger, A. 1989. *'Optimal Replication of Contingent Claims under Transaction Costs'*, Review of Futures Market 8, pp222–239.
Davis, M. H. A., Panas, V.G., and Zariphopoulou, T. 1993. *'European Option Pricing with Transaction Costs'*, SIAM Journal on Control and Optimisation 31, pp470–493.

After making adjustments for new share issues, mergers and acquisitions, listings, de-listings, and other changes, the Ask and Bid prices for the index were estimated as follows;

$$\tilde{P}_{FTSE}^{Ask} = \sum_i \left[p_i^{Ask} \frac{(q_i p_i^{Ask})}{\sum_i (q_i p_i^{Ask})} \right] \tag{4.2}$$

and

$$\tilde{P}_{FTSE}^{Bid} = \sum_i \left[p_i^{Bid} \frac{(q_i p_i^{Bid})}{\sum_i (q_i p_i^{Bid})} \right] \tag{4.3}$$

In equations (4.2) and (4.3) $\tilde{P}_{FTSE}^{Ask}$ and $\tilde{P}_{FTSE}^{Bid}$ are the estimated ask and bid prices for the FTSE 100 index. p_i^{Ask} and p_i^{Bid} are the ask and bid prices of the i^{th} stock in the index, and q_i is the number of shares in issue for the i^{th} stock. In the fractional terms within the brackets, the numerator is the market value of the ask or bid, for the i^{th} stock, and the denominator is the aggregate market value of the index for the ask or bid, respectively. The subscript i denotes a stock comprising the index and $i \in [1,100]$. There were a total of 14,381 observations for calls and 14,063 for puts. These were randomly partitioned into training sets and test sets. For the calls, the training set contained 7,138 observations and the test set contained 7,243 observations. For the puts, the training set contained 7,010 observations, and the test set contained 7,053 observations. Table 19 gives descriptive statistics of this data[95].

The option prices were separately modelled using the estimated ask and bid prices for the index. First, for both the calls and the puts, a MLP model was trained using the ask A in place of the mid-price S, as an input to the models. Then, models substituting the bid B for the mid-price S were trained, for both the calls and puts. The usual BS input variables were used for the four remaining inputs in all the models. Paired t-tests were performed using the test sets, comparing residuals of models with the ask as an input, with those of models using the bid as an input. If significant differences were found in the residuals, it would strongly suggest that the bid-ask spread on the underlying asset

[95] Inspection of Table 19 reveals that the mean daily instrument settlement price S, does not lie between the mean ask and bid prices. S is not the 1610 price of the spot index. It is the price of an implied future, with the same maturity as the option, written on the spot index. This is equivalent to the FTSE 100 spot index level at 1610. There is no significant difference in the means of the implied futures and the spot index, but the implied future has a slight upward bias compared to the spot index.

Table 19. Descriptive Statistics of the Dataset

	C/P	S	X	r	t	IV	Ask	Bid
Calls: Training Set								
Mean	89.38	3422.74	3435.1	0.06	0.2	0.14	3417.5	3398.11
Std. Devia	105.78	503.86	521.87	0.01	0.21	0.03	506.48	507.59
Range	977	2232	2700	0.07	1.01	0.34	2164.07	2162.53
Minimum	1	2284	2125	0.05	0	0	2288.68	2273.32
Maximum	978	4516	4825	0.12	1.01	0.34	4452.76	4435.84
Count	7138	7138	7138	7138	7138	7138	7138	7138
Calls: Test Set								
Mean	87.92	3428	3444.9	0.06	0.2	0.14	3422.42	3403.06
Std. Devia	105.81	493.44	509.31	0.01	0.21	0.03	494.96	496.08
Range	1113	2207	2700	0.07	1.01	0.34	2164.07	2162.53
Minimum	1	2284	2125	0.05	0	0	2288.68	2273.32
Maximum	1114	4491	4825	0.12	1.01	0.34	4452.76	4435.84
Count	7243	7243	7243	7243	7243	7243	7243	7243
Puts: Training Set								
Mean	58.19	3397.81	3304.26	0.07	0.2	0.14	3392.59	3373.19
Std. Devia	66.78	507.03	492.29	0.01	0.21	0.03	509.24	510.36
Range	661	2232	2700	0.07	0.99	0.25	2164.07	2162.53
Minimum	1	2284	2125	0.05	0	0.07	2288.68	2273.32
Maximum	662	4516	4825	0.12	0.99	0.32	4452.76	4435.84
Count	7010	7010	7010	7010	7010	7010	7010	7010
Puts: Test Set								
Mean	58.09	3398.5	3304.49	0.07	0.2	0.14	3393.28	3373.83
Std Deviat	65.48	509.81	497.67	0.01	0.21	0.03	511.39	512.5
Range	739	2232	2700	0.07	0.99	0.25	2164.07	2162.53
Minimum	1	2284	2125	0.05	0	0.07	2288.68	2273.32
Maximum	740	4516	4825	0.12	1	0.32	4452.76	4435.84
Count	7053	7053	7053	7053	7053	7053	7053	7053

Table 19. C or P is call or put price, S is the daily settlement price of the underlying asset, X is the exercise price, r is the risk free interest rate, t is the time to maturity in years, IV is the LIFFE tabulated at-the-money implied volatility, Ask and Bid are the estimated ask and bid prices of the underlying asset. Prices are expressed in index points.

cannot be ignored, and that option pricing models using the mid-price as an input are mis-specified. The results of these tests are presented in Table 20. The calculated t-statistics reported in Table 20 are greater than the critical value by a substantial margin. Hence, the difference between the ask and bid prices of the underlying asset is indeed significant for the options on the FTSE 100 for this data.

Table 20 Ask and Bid for Underlying Asset: Comparison of Models

Options	Models	Sample	Topology	R^2	F stat	t-stat	Paired t-test
	[f_{ask} v. f_{bid}]	n	[no. nodes]		[F_{crit}=0.96]	$t_{(2,0.05)}$ [t_{crit}(2 tail)=1.96]	[H0: $e^*_{Ask} = e^*_{Bid}$]
Calls	C^*(Ask,X,t,r,IV) v. C^*(Bid,X,t,r,IV)	7243	5-11-1	0.881	0.98	-8.66	Reject H0:
Puts	P^*(Ask,X,t,r,IV) v. P^*(Bid,X,t,r,IV)	7053	5-11-1	0.890	0.91	8.03	Reject H0:

Table 20 Results of F tests and paired t-tests on the residuals of option pricing models for FTSE 100 index ESX options, using the ask and bid prices of the underlying asset, respectively, as inputs, rather than the mid price. The calculated values of the t-statistic are substantially in excess of the critical value suggesting that the difference between the ask and bid prices of the underling asset is statistically significant for option prices.

This suggests that transaction costs involved in trading the underlying asset should not be ignored in a properly specified option pricing model. But how should these costs enter the model? Consider an investor who writes (sells) a call option. The hedged short position in the call option is as follows[96];

$$\begin{aligned} &- \text{Call} \\ &+\Delta \text{ shares of underlying asset} \end{aligned} \tag{4.4}$$

The writer sells one call option. To hedge against price movements of the underlying asset, he must purchase Δ shares of that asset, where delta is the partial derivative of the call pricing function with respect to the price of the underlying asset. The price he must pay to purchase the underlying asset is the *ask* price, *not* the mid price. Similarly, the hedged short position for an investor who sells a put option is;

$$\begin{aligned} &- \text{Put} \\ &-\Delta \text{ shares of underlying asset} \end{aligned} \tag{4.5}$$

The writer sells one put option. To hedge the position he must sell Δ shares of the underlying asset. The price he will receive is the *bid* price, *not* the mid price. Therefore, call options should be hedged and priced using the *ask* price of the underlying asset, and put options should be hedged and priced using the *bid* price of the underlying asset. This is because the ask and bid prices of the underlying asset and *not* the mid price are the *prices actually paid* by investors to hedge call and put options, respectively. Use of the ask and bid prices as described, rather than the mid price, should therefore yield improved pricing performance. The benchmark BS model was used to test this hypothesis. Results are presented in Table 21.

Table 21 Pricing with BS Using Ask and Bid Prices of Underlying Asset

Options	Models	Sample	Adj.R^2	F stat	t-stat	t-test
		n		$[F_{crit}=0.96]$	$t_{(2,0.05)}$ $[t_{crit}(2\text{ tail})=1.96]$	[H0: $C=C^*$/ $P=P^*$]
Calls	$C^*_{BS}(S,X,t,r,IV)$	7243	0.985	0.84	-3.35	Reject H0:
Calls	$C^*_{BS}(Ask,X,t,r,IV)$	7243	0.995	0.91	-1.6	Accept H0:
Calls	$C^*_{BS}(Bid,X,r,t,IV)$	7243	0.994	0.98	3.74	Reject H0:
				$[F_{crit}=1.04]$		
Puts	$P^*_{BS}(S,X,t,r,IV)$	7053	0.935	1.32	14.99	Reject H0:
Puts	$P^*_{BS}(Ask,X,t,r,IV)$	7053	0.952	1.29	14.17	Reject H0:
Puts	$P^*_{BS}(Bid,X,t,r,IV)$	7053	0.971	1.13	8.10	Reject H0:

Table 21 Pricing FTSE 100 index ESX call and put options with the BS formula, using the mid price and the (estimated) ask and bid prices of the underlying asset as inputs. The results indicate that the ask price provides the best fit for the calls, and the bid price provides the best fit for the puts.

Table 21 shows that for the call options, using the ask price of the underlying asset as an input to the BS formula gives an unbiased estimate of actual market prices, and the best

[96] Hull (2000), Chap 13 pp312-313.

fit as measured by Adj. R^2. Use of the mid price or the bid price produces biased estimates, and poorer fits. For the put options, all three estimates of actual market prices remain biased, as indicated by the values of the calculated t-statistics. However, using the bid price of the underlying as an input gives the smallest value of the t-statistic, and the best fit as measured by Adj. R^2. These results support the hypothesis that transactions costs in trading the underlying asset, should be accounted for in parametric option pricing models by using the ask price as an input for pricing and hedging call options, and the put price as an input for pricing and hedging put options. The results in Table 21 suggest that adjusting for transactions costs on the underlying asset, can improve the pricing performance of the BS model. Table 22 presents comparable results for pricing the same options with MLP option pricing models.

Table 22 Pricing with MLPs Using Ask and Bid Prices of Underlying Asset

Options	Models	Sample	Topology	Adj.R^2	F stat	t-stat	t-test
		n	[no. nodes]		[F_{crit} =1.04]	$t_{(2,0.05)}$ [t_{crit}(2 tail)=1.96]	[H0: C=C*/ P=P*]
Calls	**C*(S,X,t,r,IV)**	7243	5-11-1	0.998	1.01	-0.05	Accept H0:
Calls	**C*(Ask,X,t,r,IV)**	7243	5-11-1	0.996	1.02	-0.16	Accept H0:
Calls	**C*(Bid,X,r,t,IV)**	7243	5-11-1	0.996	1.02	0.02	Accept H0:
Puts	**P*(S,X,t,r,IV)**	7053	5-11-1	0.997	1.01	0.00	Accept H0:
Puts	**P*(Ask,X,t,r,IV)**	7053	5-11-1	0.995	1.03	0.10	Accept H0:
Puts	**P*(Bid,X,t,r,IV)**	7053	5-11-1	0.994	1.03	-0.09	Accept H0:

Table 22 Pricing FTSE 100 index ESX call and put options with MLPs, using the mid price and the (estimated) ask and bid prices of the underlying asset as inputs. The results suggest that a MLP is able to determine the mapping to the option prices regardless of which of the three prices for the underlying asset is used.

The results in Table 20 suggest transaction costs on the underlying asset are statistically significant for option prices and should be included in a properly specified model. However, Table 22 shows that for an MLP option pricing model in practice, there is a negligible difference in the quality of fit obtained regardless of whether the mid, ask, or bid prices of the underlying asset are used as inputs. Here, the calculated t-statistics are uniformly small, and there is no significant difference between the actual market prices of the options and the estimated prices. These results indicate that adjusting for transaction costs on the underlying asset is not critical in the case of MLPs. However, the results in Table 21 show this is not the case for the BS model, where adjustment for transaction costs in the manner suggested above produces marked improvements in performance. Overall, the results demonstrate the flexibility of non-parametric computational knowledge discovery techniques such as neural nets, when compared to parametric methods such as BS, for option pricing.

4.2.5 Conclusions

Regression techniques can be used in two ways, to discover functions and to fit functions. Conventional parametric regression techniques start out with a pre-defined function, and this is placed in the data space to best fit the data. The advantage of non-parametric computational knowledge discovery techniques such as neural nets for regression, is that no particular function is assumed. Instead, the function is determined by the data. However, in order to obtain useful results, statistically rigorous data mining model search methods must be used to select the variables to be included in the model. In addition, a trade-off in using a non-parametric modelling methodology is that a method of selecting the topology, architecture, or bandwith, (degrees of freedom) is needed. In section 4.2.2 a data mining model search method, the GeTS Model Search Algorithm for Input Space Search was demonstrated. A computational knowledge discovery technique the MLP, was used in the context of this statistically principled model search method, to show that correctly specified option pricing models should include extra variables namely, spread, volume, and open interest (*OI*). In section 4.2.3 the use of a statistically based architecture selection algorithm to choose a model topology that minimises the expected squared error, or prediction risk (*PR*), on test data was demonstrated. A method of estimating *PR* was demonstrated which is applicable when target values are unknown. In section 4.2.4. use of a computational knowledge discovery technique, the MLP, as an investigative tool to fit functions and perform a statistical test, rather than discover functions, was demonstrated. The MLP was used to model option prices, using the estimated ask and bid prices of the underlying asset as inputs in place of the mid price. The object was to determine whether transactions costs incurred in trading the underlying asset actually affected option prices. Evidence suggesting that these costs should be included in a correctly specified model for option prices was presented. Further results were presented suggesting that for the BS and similar option pricing models, call options should be hedged (priced) using the ask price of the underlying asset, and put options should be hedged (priced) using the bid price, rather than the customary practice of simply using the mid price. However, results for neural nets or equivalent techniques, suggest any of these inputs will produce predicted prices that are unbiased estimates of the observed prices.

4.3 Application II: Option Implied Probability Distributions

Market practitioners and researchers have long sought ways to extract information on future asset prices from current prices quoted in financial markets. For example, prices of forward contracts have been used to make point estimates of future asset price levels, and option prices have been used to obtain estimates of future asset price volatility. In recent years there has been increasing interest in implied probability distributions for the price of the underlying asset at expiration, estimated from option price series observed in the market. It is reasoned that quoted prices of options impound the market's expectations regarding the value of the underlying asset at the expiration of the option. It is possible to extract not just the mean, but a complete probability distribution for the future price of the underlying asset, using observed option prices. Useful information contained in the shape of the distribution can thus be recovered. This distribution is often referred to as the risk-neutral distribution (RND) and that term is used here[97]. This section deals with the application of computational knowledge discovery techniques to the extraction of the RND from market prices of options.

4.3.1 Risk Neutral Distributions: Theory and Applications

Cox and Ross (1976)[98] showed that the prices of European exercise options could be expressed as the expected value of their payoffs, discounted at the risk-free interest rate.

$$\begin{aligned} C(X,t,T) &= e^{-r(T-t)} \int_{X}^{\infty} P(S_T)(S_T - X)dS_T \\ P(X,t,T) &= e^{-r(T-t)} \int_{-\infty}^{X} P(S_T)(X - S_T)dS_T \end{aligned} \tag{4.6}$$

In equation (4.6) $C(X,t,T)$ and $P(X,t,T)$ are the prices of calls and puts trading at time t for expiration at some later time T. X is the strike price, and r is the risk-free interest rate. $P(S_T)$ is the RND for the value of the underlying asset S at time t. Given an assumption about the functional form of $P(S_T)$ options can be priced for any value of exercise price X. Conversely, given a series of synchronous market prices observed at some time t, for options expiring at some later time T, this calculation can be inverted and an estimate of $P(S_T)$ extracted. Breeden and Litzenberger (1978)[99] showed that the

[97] More rigorously it is the state price density (SPD). See e.g. Cont, R. 1997. *"Beyond Implied Volatility"*, in Keretz, J. and Konder, I. (Eds.) *Econophysics*, Klewer, Amsterdam.

[98] Cox, J.C., and Ross, S.A. 1976. *"The Valuation of Options for Alternative Stochastic Processes"*, Journal of Financial Economics 3, p 145-166

[99] Breeden, D. T., and Litzenberger, R. H. 1978. *"Prices of State-contingent Claims Implicit in option Prices"*, Journal of Business 4, p 621-651.

cumulative density function (negatively signed) for the value of the underlying asset S at time t is given by the first partial differential with respect to X of equation (4.6),

$$\frac{\partial f(X,t,T)}{\partial X} = -e^{-r(T-t)} \int_X^\infty P(S_T) dS_T \tag{4.7}$$

and the RND is obtained by differentiating equation (4.6) twice with respect to X.

$$\frac{\partial^2 f(X,t,T)}{\partial X^2} = e^{-r(T-t)} P(X). \tag{4.8}$$

In equations (4.7) and (4.8) $f(X,t,T)$ represents the call or put option pricing functions. To understand why equation (4.8) gives the RND consider the portfolio known as a 'butterfly spread'. This is given by

$$C(X+\varepsilon,t,T) - 2C(X,t,T) + C(X-\varepsilon,t,T) \tag{4.9}$$

in equation (4.9), ε is a small increment. The portfolio is created by selling two call options at exercise price X, and by purchasing a single call option at exercise price $(X + \varepsilon)$ and another at $(X - \varepsilon)$. Except in the interval $[X - \varepsilon, X + \varepsilon]$ the portfolio makes no payout. Consider $1/\varepsilon^2$ shares of this portfolio; in the limit as ε tends to zero, the payoff function tends to a Dirac delta function with mass at X, thus the portfolio will pay £1/$1 if $S_T = X$ and nothing otherwise. The price of the portfolio (4.9) must be

$$\frac{1}{\varepsilon^2}[C(X+\varepsilon,t,T) - 2C(X,t,T) + C(X-\varepsilon,t,T)] \tag{4.10}$$

and

$$\underset{\varepsilon \to 0}{Lim}\left(\frac{1}{\varepsilon^2}[C(X+\varepsilon,t,T) - 2C(X,t,T) + C(X-\varepsilon,t,T)]\right) = \frac{\partial^2 C(X,t,T)}{\partial X^2} \tag{4.11}$$

Thus the risk-neutral probability that $S_T = X$ is the price of a butterfly spread centered at X, in the limit as $\varepsilon \to 0$, and this is equal to $e^{-r(T-t)} p(X)$. In reality, X is not continuous and options are only available for a limited number of exercise prices at discrete intervals. However, Breeden and Litzenberger (1978) have shown that for discrete data, finite difference methods can be used to obtain a numerical solution to equation (4.8). In addition, Neuhaus (1995)[100] has shown how the RND can be obtained via equation (4.7) using finite differences.

RNDs have many practical applications. They are used by central banks to assess market expectations regarding future stock prices, commodity prices, interest rates, and

[100] Neuhaus, H., 1995. "*The Information Content of Derivatives for Monetary Policy*", Discussion Paper 3/95, Economic Research Group of the Deutsche Bundesbank.

exchange rates, in connection with setting monetary policy[101]. They are useful to market practitioners as an aid to investment decisions. RNDs extracted from exchange traded options can be used to price exotic options. For risk management, they can provide measures of value-at-risk (VaR)[102].

4.3.2 Recovering RNDs: Existing Methods

A variety of approaches has been developed for estimating probability density functions from option prices including:

a) Recovery of the assumed stochastic process for the price of the underlying asset, with the RND obtained as a by-product[103].
b) Assumption of some functional form for the RND and estimation of its parameters by minimising the difference between actual and predicted option prices[104].
c) Smoothing techniques that relate option prices empirically to exercise prices, allowing recovery of the RND through differentiation[105].
d) Non-parametric techniques such as kernel regression[106].

Techniques of type a) and b) work with equation (4.6) using assumptions about the stochastic price process for the underlying asset or the RND itself, and evaluation of the integral to estimate its parameters so predicted option prices best fit observed option prices. Techniques of type c) and d) use equations (4.7) or (4.8) and the option pricing function $f(X,t,T)$ is differentiated (numerically or analytically) to obtain the RND[107]. However, two techniques have been adopted as standard by practitioners. Namely, the mixture of lognormals, and the smoothed implied volatility smile. The first of these is a

---

[101] See e.g. Bahra, B. 1997. *"Implied Risk-Neutral Probability Density Functions From Option Prices: Theory and Applications"*, Bank of England, Threadneedle Street, London, EC2R 8AH.
[102] Ait-Sahalia, Y., and Lo, A.W. 2000. "*Nonparametric Risk Management and Implied Risk Aversion*", Journal of Econometrics 94, pp 9-51.
[103] Maltz, A.M. 1996. *"Using Option Prices to Estimate Ex-Ante Realignment Probabilities in the European Monetary System: the case of sterling-mark"*, Journal of International Money and Finance, 15, pp 717-48.
[104] Ritchey, R.J. 1990. *"Call Option Valuation for Discrete Normal Mixtures"*, Journal of Financial Research 13, pp285-296.
[105] Shimko, D. 1993. *"Bounds of Probability"*, Risk, Vol.6, No.4, pp33-37.
[106] Ait-Sahalia, Y., and Lo, A.W. 1998. *"Nonparametric Estimation of State Price Densities Implicit in Financial Asset Prices"*, Journal of Finance, Vol. LIII No2, pp499-547.
[107] For a recent review of these techniques see Jackwerth, J. C. 1999. *"Option Implied Risk-Neutral Distributions and Implied Binomial Trees: A Literature Review"*, Journal of Derivatives,Winter 1999, pp 66-82.

parametric method of type b) which works with equation (4.6); the second is a non-parametric method of type c) which works with equations (4.7) or (4.8).

The mixture of lognormals technique originated with Ritchey (1990). In this method, the RND is represented by the weighted sum of two or more independent lognormal density functions.

$$P(S_T) = \sum_{i=1}^{n} \left[\theta_i LnD(\alpha_i, \beta_i; S_T) \right] \tag{4.12}$$

In equation (4.12), $LnD(\alpha_i, \beta_i; S_T)$ is the i^{th} lognormal density function for the asset price at maturity S_T, in the n component mixture with parameters α_i and β_i. The parameter θ_i is a probability weighting and must satisfy the condition

$$\sum_{i=1}^{n} \theta_i = 1, \quad \theta_i > 0 \forall i \tag{4.13}$$

This representation has the advantage that it offers greater flexibility than a single density representation. In principle, a mixture of Gaussian (or other) densities can approximate any continuous density to arbitrary accuracy as the number of component density functions tends to infinity[108]. Ritchey however used a mixture of two lognormal densities to minimise the number of parameters to be estimated, and this has become the standard procedure. If equation (4.12) is substituted into equation (4.6) the resulting expressions can be fitted to observed call and put prices, and the parameters estimated to minimise the weighted sum of fitted price errors, using non-linear optimisation methods. Given the parameters *and* the observed option prices, the implied RND can then be constructed. The mixture of lognormals method is computationally intensive and the number of parameters to be estimated increases rapidly with the number of density functions included. Since option price series frequently have 20 or fewer observed prices corresponding to different exercise prices, this means the method is prone to overfitting.

The smoothed implied volatility smile method originated with Shimko (1993). The RND can be obtained directly from equation (4.8) provided the option pricing function $f(X,t,T)$ is observable. Unfortunately, only a relatively small number of option prices corresponding to discrete exercise prices are observable for a given time t. An obvious

[108] Titterington, D.M., Smith, A.F.M., and Makov, U.E. 1985. *Statistical Analysis of Finite Mixture Distributions*, John Wiley: New York.

solution is to smooth and interpolate the observed prices by fitting a function to them. Shimko considered this approach, but found that the smoothing spline functions he used were not suitable for fitting option prices. To overcome this difficulty Shimko converted the prices to implied volatilities using the BS formula. He then fitted a quadratic polynomial smoothing function to the available implied volatilities and used linear extrapolation outside the range of observable exercise prices. The continuous implied volatilities obtained were then converted back to continuous option prices using BS, and a RND extracted using the relation in equation (4.8). In this method, the BS formula is used purely to effect a transformation from one data space to another (option price/exercise price to implied volatility/exercise price). The method does not rely on any of the assumptions underlying the BS formula. The method has been refined by Maltz (1997)[109] and others. The smoothed volatility smile method has the advantage that fitting polynomial curves or splines to the smile can be done in a single pass, without iteration. However, the probabilities obtained in extracting the resulting RND cannot be guaranteed to be positive as this is not a constrained optimisation, and this needs to be separately checked.

Bliss and Panigirtzoglou (2002) and Andersson and Lomakka (2003) have carried out comparative performance tests on the two methods[110]. Bliss and Panigirtzoglou found that the smoothed implied volatility smile estimation method produced more stable RNDs than the mixture of lognormals method. It was found remarkably free of computational problems, and reasonably insensitive to small measurement errors. By contrast, the mixture of lognormals method was found to be sensitive to computational problems, suffered from frequent convergence failures, and had a tendency to produce spurious spikes in estimated RNDs. They suggest that despite its popularity, the mixture of lognormals method should not be used. Moreover, Bliss and Panigirtzoglou suggest the smoothed implied volatility smile method may be improved by trading goodness-of-fit for a more stable RND, and point out that the mixture of lognormals method does not permit this fine tuning. The authors warn though, that skewness and kurtosis cannot always be precisely estimated with either method, particularly for price

[109] Malz, A.M. 1997. *"Estimating the probability distribution of the future exchange rate from options prices"*, Journal of Derivatives 5, pp 20-36.

[110] Bliss, R., and Panigirtzoglou, N. 2002. *"Testing the Stability of Implied Probability Density Functions"*, Journal of Banking and Finance 26, 381-422.
Andersson, M., and Lomakka, M., 2003. *"Evaluating Implied RNDs by Some New Confidence Interval Estimation Techniques"*, Sveriges Riksbank Working Paper Series, No. 146.

series with small numbers of observed option prices. Andersson and Lomakka also concluded the smoothed implied volatility method was superior in performance to the mixture of lognormals. They used the width of confidence intervals, estimated using bootstrap and Monte Carlo methods, as a criterion for evaluating the two methods, and found that the smoothed implied volatility smile method yielded tighter confidence intervals for estimated RNDs.

4.4 Recovering RNDs for FTSE 100 Index American Put Options

The theory underlying RNDs is only applicable to European exercise options, and cannot be applied to American exercise options without modification. However, most exchange traded options are actually American options. American options can be exercised at any time prior to maturity, and it is reasonable to suppose that this feature, which is reflected in their price, affects any extracted RND. It is an open question whether RNDs extracted from American options are significantly different empirically to those extracted from corresponding European options. If there is no significant difference, then American options could be used interchangeably with European options for extracting RNDs. If there is a significant difference, it raises questions as to how that difference should be interpreted, and how and to what extent RNDs recovered from American options can be used in practice.

In this section, the above questions are used as a case study to demonstrate the application of computational knowledge discovery techniques (specifically MLPs) to the recovery of RNDs from option prices. Only put options are considered. This is because early exercise of American call options is never optimal[111]. Therefore, American call options can be priced as European call options. In this case study, RNDs from American and European put option pricing functions are extracted and compared.

4.4.1 Motivation

The non-parametric extraction of RNDs from option prices is an example of an ill-posed problem, in that small changes in option prices can lead to large changes in the estimated density function. It requires the use of methods that are robust to slight changes in the option prices. One solution is to impose some degree of regularization

[111] In the absence of dividend payments. See e.g. Hull(2000), Chapter 7 p175.

(smoothing) on the data[112]. In the smoothed implied volatility smile method this is achieved by fitting the data in the implied volatility/exercise price space. This is necessary because the quadratic polynomial and smoothing spline functions used to fit the data in that method, proved unsuitable for directly fitting option prices, with their exponential-like functional form, asymptotic to a slope of zero for deep out of the money options, and a slope of one for deep in the money options. In contrast, neural nets and other sufficiently general computational knowledge discovery techniques, have been shown to be suitable for directly fitting option prices[113]. The required smoothing can be controlled by the choice of bandwith, degrees of freedom, or number of hidden layer nodes[114]. Moreover, as noted by Rebonato (1999)[115] fitting directly to option prices is an improvement on the (in principle) least data-polluting procedure of using only the actually quoted option prices. For this work, RNDs were extracted by first estimating smoothed prices corresponding to each exercise price in a daily price series, by directly fitting MLP option pricing functions, and then twice differentiating the functions numerically[116], partially with respect to exercise price.

Surprisingly, there has been little study of the use of neural nets for extracting RNDs from option prices. A version of the parametric mixture of lognormals method was implemented by Schittenkopf and Dorffner (2000)[117] using Mixture Density Networks[118]. Herrmann and Narr (1997) differentiated NN option pricing functions fitted directly to prices of options on the German DAX index to obtain RNDs. They used average values for some input variables when training their models. The resulting RNDs were compared graphically with corresponding lognormal RNDs obtained using the BS formulae. No statistical tests were performed, and only goodness-of-fit and error measures were provided. Despite an extensive literature search, no other studies were found.

---

[112] Tikhonov, A.N., and Arsenin, V.L. 1977. *Solutions of Ill-Posed Problems*, Scripta Series in Mathematics, Halstead Press, Winston, New York.

[113] See e.g. Hutchinson, Lo and Poggio (1994) and Galindo (1999).

[114] This approach is *structural regularisation* as opposed to *formal regularisation* as defined in; Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L., and Hopfield, J. 1987. *"Large Automatic Learning, Rule Extraction, and Generalisation"*, Complex Systems, 1(5), pp877-922.

[115] Rebonato, R. 1999. *Volatility and Correlation*, Wiley Financial Engineering, John Wiley & Sons Ltd. Chichester. p 195.

[116] Bishop (1995) notes that both first and second partial derivatives of neural networks can be obtained by using finite differences, with accuracy limited by a computer's numerical precision.

[117] Schittenkopf, C., and Dorffner, G. 2000. *"Risk Neutral Density Extraction From Option Prices: Improved Pricing With Mixture Density Networks"*, working paper, Austrian Research Institute for Artificial Intelligence, Schottengasse 3, 1010 Vienna, Austria.

[118] Bishop (1994).

4.4.2 Data and Methodology

First, suitable data sets were created using the data preparation phases and tasks of the Computational Framework. For the European exercise options, a training set of 13,790 FTSE 100 ESX put options was created. For the American exercise options the training set contained 14,619 FTSE 100 SEI put options. To ensure prices of liquid options were used, only options with positive values for contract volume and open interest were selected. A disjoint test set of 60 daily option price series, containing data on a total of 1,238 (European) put options on the FTSE 100 index, was created. Underlying asset prices for American options, which differ from those for European options, were also added. This test set had the following two special features:

1) The options included were trading for one of 60 consecutive available monthly expirations.
2) The options had a maturity of one calendar month (17 or 18 trading days). This figure was selected because any longer maturity resulted in overlapping data for some variables.

In creating the test set the object was to obtain a set of option price series with constant maturities, which was non-overlapping. The latter feature was required to avoid serial dependence between successive observations, which might bias statistical results for the RNDs[119]. Pricing models for European and American put options were separately trained, using a 5-11-1 architecture. The inputs were the five BS variables, and the targets were market prices of European and American put options, respectively. Once trained, the pricing models were applied to the test set to generate series of smoothed European and American option prices, taking care to use the correct values of the underlying asset as inputs to each model. Each generated price series was then numerically differentiated to estimate $\partial^2 f(X,t,T)/\partial X^2$ using symmetric central finite differences. The following formula was applied.

$$\frac{\partial^2 f(X,S,t,r,\sigma)}{\partial X^2} \approx \frac{f(X+\varepsilon,S,t,r,\sigma)-2f(X,S,t,r,\sigma)+f(X-\varepsilon,S,t,r,\sigma)}{\varepsilon^2} \quad (4.14)$$

[119] Methods exist for handling overlapping data, which can lead to error reductions in calculated statistics in some cases. See e.g. Dacorogna et al (2001), pp 47-51. However, this is at the cost of considerable computational complexity, so overlapping data was not used here.

where $f(X,S,t,r,\sigma)$ is a neural net option pricing function with the 5 standard BS input variables, and ε is a small increment. If $\hat{p}(X_i)$ is the estimated probability density over an interval $[X_i - 0.5\varepsilon, X_i + 0.5\varepsilon]$, then

$$\hat{p}(X_i) \approx \frac{\partial^2 f(X_i,S,t,r,\sigma)}{\partial X^2}\varepsilon \tag{4.15}$$

where X_i is the i^{th} observable exercise price (or class middle) in a series of n, and ε is the interval between adjacent values of X_i. Equation (4.15) was used to obtain point estimates of the probability density corresponding to each X_i. The median value, and probabilities in the tails of the distribution below the 1^{st} and above the n^{th} observable option price in the series, were estimated as suggested by Neuhaus (1995), using the relation in equation (4.7). Finally, the first four moments of each recovered RND were estimated as follows:

$$Mean = \sum \frac{X_i + X_{i+1}}{2}\hat{p}(X_i) \tag{4.16}$$

$$Stdev = \sqrt{\sum\left(\frac{X_i + X_{i+1}}{2} - Mean\right)^2 \hat{p}(X_i)} \tag{4.17}$$

$$Skewness = \sum\left(\frac{X_i + X_{i+1}}{2} - Mean\right)^3 \frac{\hat{p}(X_i)}{Stdev^3} \tag{4.18}$$

$$Ex - Kurtosis = \sum\left(\frac{X_i + X_{i+1}}{2} - Mean\right)^4 \frac{\hat{p}(X_i)}{\left(Stdev^4 - 3\right)} \tag{4.19}$$

The annualised percentage implied volatility was also calculated from equation (4.17) for each series. The resulting sets of summary statistics for the RNDs are given in Appendix E.

4.4.3 Comparisons of RNDs of European and American Put Options

Table 23 gives results for direct comparisons of the time series of summary statistics obtained for the European and American put options. The results of the paired t-tests suggested that significant differences exist between RNDs for European and American put options. In particular, the results for skewness and excess kurtosis, suggested the shapes of the RNDs were different for each type of put option.

Table 23 European v. American FTSE 100 Index Options: Comparison of RNDs.

Summary Stats.	R^2	F-stat.		t-stat (paired).		Paired t-test
[European v. American]		Fcrit	Fcalc(0.05)	t_{crit}(2 tail)	t_{calc}(2,0.05)	[H0:=no difference]
Median	0.999	1.54	1.00	2.00	5.07	Reject
Ann.I.V.%	0.820	1.54	1.78	2.00	4.29	Reject
Mean	0.999	0.65	0.99	2.00	9.09	Reject
S.Deviation	0.640	1.54	1.45	2.00	5.35	Reject
Skewness	0.932	0.65	0.92	2.00	19.01	Reject
Ex.-Kurtosis	0.779	0.65	0.51	2.00	-13.04	Reject

Table 23 Comparison of time series of summary statistics for RNDs recovered from non-overlapping, constant maturity (17/18 trading days), sets of European and American exercise put options on the FTSE 100 Index. The test data set contained 1,238 put options in 60 daily option price series for 60 consecutive monthly expirations. The results of the paired t-tests suggested there are statistically significant differences between RNDs for European and American put options, particularly for the higher moments.

Further tests were carried out to assess the practical effects of these differences on the predictive properties of RNDs from each type of put option. In these tests, results of which are presented in Table 24, the median[120] and annualised % implied volatility from each RND, are compared with the actual FTSE 100 closing price (T.FTSE 100) and realised volatility[121] on the expiration date of the option. This is a test of the one month (18 trading day) forecasting performance of the estimated RNDs.

Table 24 One Month Forecast Performance: Comparison of RNDs

Parameter	R^2	F-stat.		t-stat		t-test
[Actual v. Forecast]		Fcrit	Fcalc(0.05)	t_{crit}(2 tail)	t_{calc}(2,0.05)	[H0:=no difference]
RNDs From FTSE 100 Index ESX European exercise options						
T.FTSE 100 v. Median	0.955	0.65	0.99	1.98	-0.02	Accept
Realised Vol.v. Ann.I.V.%	0.379	1.54	2.01	1.98	-3.47	Reject
RNDs From FTSE 100 Index SEI American exercise options						
T.FTSE 100 v. Median	0.955	0.65	0.99	1.98	0.05	Accept
Realised Vol.v. Ann.I.V.%	0.318	1.54	3.58	1.99	-2.55	Reject
Memorandum Item						
Realised Vol. v. ATMIV(LIFFE)	0.426	1.54	1.46	1.98	-4.50	Reject

Table 24 Tests of the predictive abilities of the median, and annualised implied volatility, of RNDs from each type of option for FTSE 100 closing prices and realised (historical) volatilities at expiration of the options, one month (17/18 trading days) later. The t-test results show that for both European and American exercise options, an unbiased estimate of the FTSE 100 closing price is obtained. In both cases, the estimate of the realised volatility is biased. However, the estimate of realised volatility provided by LIFFE tabulated at-the-money implied volatility, seen in the bottom line, is even more biased.

The t-test results in Table 24 indicate that the medians of RNDs from both American and European exercise options provide an unbiased estimate of FTSE 100 closing prices on the expiration date of the options, one month (18 trading days) later. The annualised implied volatilities of the RNDs from both types of options, on the other hand, are

[120] RNDs are not symmetrical, the median is the correct statistic to use for non-symmetrical distributions because 50% of a distribution is above the median and 50% is below. Hence, the FTSE 100 closing price on the expiration date of the option is equally likely to be above or below the median.

[121] The historical volatility estimated from the time series of FTSE 100 returns for the 18-day period up to and including each expiration date. This time period matches the 18-day maturity of the options.

biased estimates of the actual realised volatilities at expiration of the option. However, the volatility estimates from the RNDs compare favourably with LIFFE tabulated at-the-money implied volatility, which gives an even more biased estimate of realised volatility. These results suggest unbiased estimates of future asset prices can be obtained from RNDs from both European and American put options on those assets.

4.4.4 Conclusions

The results presented in Table 24 are surprising. They suggest that in practice, RNDs from both European and American put options can be used interchangeably to obtain forecasts of future asset prices, contrary to theory, and the existence of significant model differences revealed by the (albeit more powerful) paired t-test results presented in Table 23. The early exercise feature does not appear to bias the forecasts of the underlying asset price at expiration obtained from the American put option RNDs. In addition, the standard deviation of RNDs for the American put options is smaller at 107.77 compared to 113.48 on average for the European options. This is counter intuitive, as it might be supposed that the early exercise feature would increase the uncertainty of the forecast. The properties of the forecasts obtained from the RNDs can be more easily appreciated by considering Fig.12 and Fig.13. The two forecasts are difficult to distinguish visually in the figures. Interestingly, in all cases the actual FTSE 100 closing prices lie within ±2 standard deviation (≈95.46%) confidence intervals, constructed from the estimated standard deviations of each separate RND.

The terminal FTSE 100 price and realised volatility are the first two moments of the realised asset price density function[122]. A few researchers, for example Anagnou et al (2002)[123], have suggested RND forecasts are incapable of producing unbiased estimates of *any* of the moments of the realised asset price density function. However, their

[122] Realised asset price densities are derived from the historical time series of asset prices and are risk-adjusted, in that the rate of drift used in calculating them is the dividend yield of the asset. In contrast RNDs are risk-neutral and use the risk-free interest rate as their drift rate. This means that the two distributions are not directly comparable. They can be rendered directly comparable by applying Girsanov's Theorem and effecting a change of measure. However, this only changes the mean, the values of higher moments are unaffected. For a discussion see e.g. Ait-Sahalia, Y. Wang, Y. Yared, F. 2001. *"Do Option Markets Correctly Price the Probabilities of Movement of the Underlying Asset?"* Journal of Econometrics, 102, pp67-110.

[123] Anagnou, I., Bedendo, M., Hodges, S., and Tompkins, R. 2002. "*The Relation Between Implied and Realised Probability Density Functions*", working paper, Financial Options Research Centre, University of Warwick, Coventry, CV4 7AL.

Fig.12 ESX Put Option RNDs: One Month Forecast of FTSE 100 Level

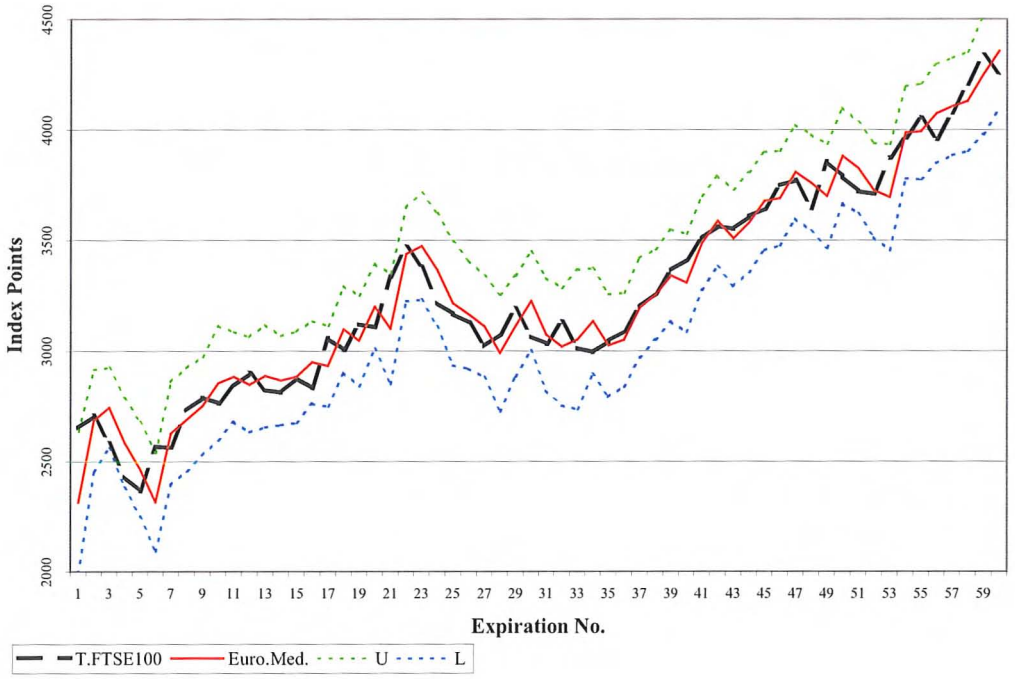


Fig.13 SEI Put Option RNDs: One Month Forecast of FTSE 100 Level

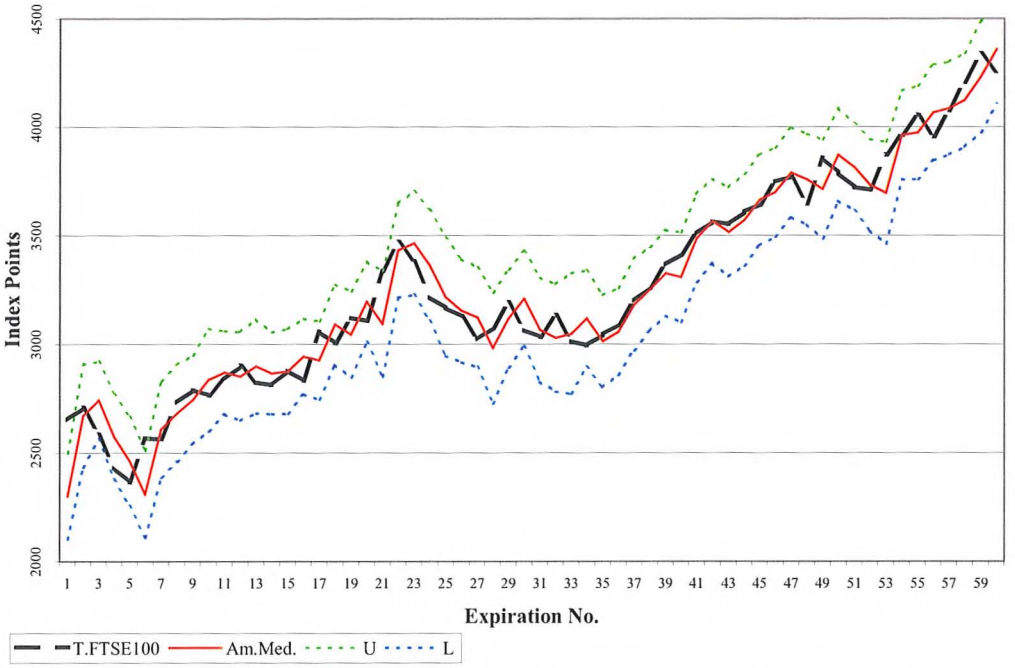


Fig.12 and 13 One month forecasts of the FTSE 100 closing price from RNDs for European and American options respectively. The forecast values (red lines) are the medians of the RNDs. The heavy black dashed lines are the FTSE 100 closing price one month (17/18 trading days) later. The dotted green lines are ±2 standard deviation (≈95.46%) confidence intervals. The true values lie within the confidence band in all cases.

conclusions are based on the use of the double lognormal or other parametric methods, and the smoothed implied volatility method. In addition to their other limitations discussed in section 4.3.2, these methods estimate RNDs directly from unsmoothed quoted option prices, some of which may relate to non-traded illiquid options[124]. This can result in biased estimates. In the method presented here, RNDs are estimated from smoothed (fitted) prices generated by neural networks, rather than actual observed quoted prices. Because the neural nets are trained only on the prices of liquid options they can produce smoothed price estimates that are consistent with liquid option prices, for any illiquid options in a price series.

For both the European and American put options, the medians of the RNDs give unbiased forecasts of the terminal FTSE price. The forecasts of realised volatility were biased, but less so than the implied volatility tabulated by LIFFE. Some caution is necessary in interpreting the latter forecasts, as it is difficult to reliably estimate realised volatility from at most 18 daily observations of returns. These results suggest that neural nets and similar computational knowledge discovery techniques of sufficient flexibility, provide a promising method for use in extracting RNDs from option prices, which merits further investigation. In addition, for finance they suggest that the early exercise feature of American options does not bias the forecast of the underlying asset price at maturity that can be obtained from the RND.

[124] Researchers sometimes estimate half of each density from out of the money calls, and the other half from out of the money puts in an attempt to circumvent this difficulty.

CHAPTER 5

Evaluation and Conclusions

5.0 Introduction

This final chapter of the thesis is in three sections. The approved scope and requirements for the research are given in Appendix H and the first section gives a summary of how these are satisfied. The second is a discussion of software implementation issues. This focuses on the nature of available software implementations of computational knowledge discovery techniques, and comments on their suitability for operational KDD / data mining operations involving options market databases. The third part closes the work by presenting conclusions and considering directions for further research, based on the results, knowledge, and practical experience, obtained in the course of the research project.

5.1 Evaluation of Results

In this section the results of the research are summarised and evaluated in their logical order. Aspects of the Computational Framework presented in Chapter 2 are discussed first. The proposed methods for estimating confidence and prediction intervals detailed in Chapter 3, are then evaluated. Next, the results reported in Chapter 4, obtained by applying elements of the Computational Framework to the case study examples, Applications I and II, are evaluated. Comments on the utility of the overall computational framework in practice follow.

5.1.1 The Computational Framework

The primary objective of this thesis was to outline a formal systems framework for the application of computational knowledge discovery techniques to options market databases. This is presented in Chapter 2. The framework is based on the industry standard CRISP-DM methodology introduced in August 2000. To create the framework the CRISP-DM Generic Process Model was systematically specialised, to obtain a process model specifically for options market applications, written in terms of CRISP-DM. First, the stages of the KDD process as defined by Fayyad et al (1996), were mapped to the Phases of the CRISP-DM ver. 1.0 Reference Model, after adjustments for minor differences in terminology and demarcations. These mapping which are illustrated in Fig.3, allowed the CRISP-DM methodology to formally encompass the full KDD process. The Phases of the CRISP-DM Reference Model in turn, were renamed to suit the Data Mining Context (Table 2), then mapped to the stages of the specialised KDD process for options markets. These are described in

Chapter 1 section 1.3.2. These further mappings, which are illustrated in Fig.5, defined the Phases of the Specialised Process Model for Options Market Applications. The specific Specialised Tasks and their outputs, for each Phase of the Specialised Process Model for Options Market Applications were then defined and detailed. These are described in Chapter 2, section 2.4. The result is a complete systems framework, compliant with the industry standard CRISP-DM methodology.

To ensure that computational knowledge discovery techniques are used in a manner that avoids attribution of significance to spurious models and predictions, a systematic model search methodology is incorporated in the framework. This is based on a synthesis of the theory of model reduction described in Hendry (1995), and the characterisation of the learning or regression problem by Moody (1994), and is described in Chapter 2, section 2.4.5. Practical application of the methodology is based on two algorithms, the GeTS Model Search Algorithm for Input Space Search (Table 3), and the Architecture Selection Algorithm (Table 4). The first of these is a shrinking algorithm, used to select the regressors for a model. The second is a growing algorithm used for architecture selection. A final feature of the model search methodology is the use of 'Bonferroni' type techniques to detemine true significance levels based on the number of hypothesis tests performed in building a model.

5.1.2 Estimation of Prediction Risk, Confidence and Prediction Intervals

A method for determining confidence and prediction intervals and estimating prediction risk, applicable to a variety of different computational knowledge discovery techniques for non-parametric non-linear regression is presented in Chapter 3. The exposition is based on neural nets (MLPs), but derivations are given in sections 3.4.1 and 3.4.2 which show that it is equally applicable to a broad class of related techniques. The motivation for developing this new method was the unsatisfactory nature of existing methods (reviewed in section 3.3); because of their computational cost, poor scalability for use with large data sets, and lack of robustness to the problems of regression typical for options market data.

The method presented is based on directly modelling the error using a special training algorithm. It simultaneously estimates the predicted value of the target variable and the variance of the predicted value, but does not have the limitations of methods using the

inverted Hessian matrix, or bootstrapping. In contrast to an otherwise similar existing method [Nix and Weigend (1995)], it deploys a standard MLP architecture, uses a least squares cost function, does not assume normality of the residuals, and uses independent training and validation sets rather than interchanging validation sets, in the training algorithm (Table 6). These, and other features, described in detail in section 3.4, mean that the method applies to any non-linear mapping of sufficient flexibility. A caveat is that the method underestimates the variance where the error is small $\left(R^2 > 0.99\right)$, but that is a limitation of all methods based on least squares and maximum likelihood estimators. The method is not computationally costly, and scales well to large data sets.

5.1.3 Results for Applications I and II

In Chapter 4 the Computational Framework and its key components were demonstrated in applications to several questions of interest in the financial options domain. Application I consisted of three applications related to option pricing. The first of these was used to demonstrate the Data Preparation Phases and the GeTS Model Search Algorithm (Section 4.2.2). It was found that correctly specified option pricing models should include extra variables to impound the information contained in transaction costs and measures of market liquidity. This result is of interest because it was obtained using a non-linear non-parametric regression technique rather than the OLS or parametric NLLS methods that are normally used for empirical tests, but are unsuitable for this data. The effectiveness of the Architecture Selection Algorithm and prediction risk criterion were tested next (Section 4.2.3), the results are discussed in the following section. Finally, the use of computational knowledge discovery techniques to fit data and perform statistical tests, outside of the context of model search, was demonstrated (Section 4.2.4). The results suggested that transaction costs involved in trading the asset underlying an option were very significant for option prices. For the BS model, there was some evidence that call options should be priced and hedged using the ask price of the underlying asset, and put options using the bid price. The pricing efficiency of neural nets was found to be such, that predicted prices indistinguishable from observed prices were obtained using either of these inputs, or the mid-price. These are new findings for finance. Application II is an interesting new application for computational knowledge discovery techniques. Namely, the non-parametric extraction of RNDs from option prices. Here, a neural net was used to produce smoothed and twice differentiable estimates of option prices. These were then numerically

differentiated to obtain estimated RNDs. In tests, the medians of RNDs from both American and European exercise put options gave unbiased estimates of FTSE 100 closing prices on the expiration date of the options, one month later. The actual FTSE 100 closing prices were all within ± 2 standard deviation confidence bands constructed from the standard deviations of the RNDs. These findings suggest that in practice, RNDs from both European and American put options can be used interchangeably to obtain forecasts of future asset prices. However, they relate to a single underlying asset, and are for a single maturity. Neural nets have not previously been evaluated in this context and these results suggest that they work well. However, a wider investigation involving different underlying assets, different maturities, and different time periods, is required for definitive conclusions.

5.1.4 Utility of the Computational Framework

The previous section focused on the findings made when the Computational Framework was applied to the case study examples. This section is concerned with the utility of the Computational Framework for obtaining those findings. A constraint in evaluating the framework was that operational deployment in a commercial enterprise was not possible. This meant that certain business related tasks in the Data Selection and Reporting phases were not included in the evaluation. Examples are; Formulate Project Plan, Recommend Deployment/Non-use, and Plan Monitoring and Maintenance. Comment is therefore confined to the data preparation, data mining, and reporting phases. The program of research as outlined in the Approved Document (see Appendix H), and the decision to use UK data from LIFFE, effectively defined the outputs for the Data Selection phase. The activities involved in data preparation were found to factor naturally into the tasks of the Data Cleaning, Data Reduction/Enrichment and Data Preparation phases, confirming that these phases of the framework were effective in practice. In the Data Mining phase, the GeTS Model Search Algorithm for Input Space Search was found to provide a workable solution to the problem of variable selection in specifying models. The search algorithm has important limitations however; Model specification tests are limited to the Jarque-Bera test, and only paired t-tests and F tests of fitted values or residuals are available for variable deletion. These limitations exist because option pricing models have residuals that are not normally distributed, which rules out the use of most other tests. Moreover, it is not normally possible to use t-tests and F tests of individual parameters for variable deletion, since these parameters are not

associated with individual input variables in neural nets and similar non parametric regression techniques.

The method proposed for the Search for Optimal Architecture task was the Architecture Selection Algorithm (Table 4) using the prediction risk criterion (*PR*). In a study of corporate bond ratings, Moody and Utans (1991)[125] showed that *PR* was an effective criterion for architecture selection. However, bond rating is a classification problem whereas option pricing is a regression problem, and model selection via prediction risk proved to be less satisfactory for the option pricing application considered here. For NNs and similar computational knowledge discovery techniques used for regression, the *PR* criterion does allow the user to choose a model with the lowest generalisation error from a set of models with different architectures (see Table 17). It does not provide a means of discovering the globally optimum architecture however. This is because these techniques require an iterative search of a weight space to minimise the cost function. However, the search algorithm can terminate in a local minimum rather than finding the global minimum. The result is that the curve of *PR* for networks with increasing numbers of hidden layer nodes, does not decline monotonically, precluding identification of the global minimum. The remaining task in the Data Mining phase is Estimate True Significance. Provided the input space search and architecture search have been properly implemented, this is a straightforward matter, and the correct values can be extracted from a look-up table such as Table 5. Finally, with respect to the Reporting Phase, this thesis itself fills the role of the Final Project Report Document, which is the output of the Produce Final Report task.

5.2 Implementation Considerations

There is an abundance of software implementations of computational knowledge discovery techniques produced by academic researchers and commercial enterprises. These range from implementations of individual techniques, to full data mining suites that include data manipulation tools and a variety of computational knowledge discovery techniques for different tasks. However, little of what is currently available is truly suited for applications to options market databases. This section explains why, and discusses current developments that may provide tools that are more suitable. Key

[125] Moody, J., and Utans, J. 1991. *"Selecting Neural Network Architectures via the Prediction Risk: Application to Corporate Bond Rating Prediction"*, Proceedings of the First International Conference on Artificial IntelligenceApplications on Wall Street. IEEE Computer Society Press, Los Alamitos, CA.

features that are essential for work with options market data, and should be included in the next generation of specialist applications for the domain are also discussed.

5.2.1 Software Implementations of Individual Techniques

Software implementations of computational knowledge discovery techniques for single tasks, for example to find clusters, or build decision trees, began to appear around 1980. These early tools originated in the academic community and were research orientated. They represent the first generation of data mining systems. The problem in using such generic individual software implementations is that they requires different data and metadata formats, have different user interfaces of varying sophistication, and lack a consistent and unifying working environment. They do not therefore provide support for a coherent KDD and data mining process. These difficulties can be partly overcome, by using a collection of software implementations written for a common operating environment. The Netlab library written for Matlab is an example. Netlab[126] is a toolbox of Matlab functions and scripts, based on Bishop (1995). It provides Matlab implementations of some of the newest machine learning algorithms. The disadvantages are that the user must be a technically sophisticated specialist. An expert knowledge of the algorithms, as well as the Matlab language and environment, is needed in order to use Netlab effectively. Moreover, a fundamental limitation is that Matlab is unable to accept categorical data. Such data therefore needs to be converted in advance, and use of techniques such as a priori and decision trees is ruled out. The level of expert knowledge needed, the limitations inherent in research orientated scientific programming tools like Matlab, and the lack of support for a KDD process, makes Netlab and similar toolboxes unsuitable for operational deployment for large scale data mining in a business enterprise.

5.2.2 Data Mining Suites

Data mining suites first appeared in the mid 1990s. They are generic toolkits for multiple tasks, with support for data pre-processing and interoperability with other applications, including DBMS. Data mining suites are mainly produced by commercial vendors, and their evolution has been vendor-driven. Currently, they offer the best solution for enterprise applications, for several reasons; The most obvious advantage of a data mining suite, is that a whole range of data manipulation tools and data mining

[126] Nabney, I.T. 2002. *Netlab: Algorithms for Pattern Recognition*, Springer Verlag, London.

algorithms are integrated. This means only one data and metadata format is needed, and different approaches can be combined and their results easily compared. Such suites feature a GUI, usually based on a process flow metaphor, which provides a consistent working environment for the data mining algorithms included, making them easy to use. They provide better support for the KDD process because they include tools for data import, cleaning, and transformation. A disadvantage of data mining suites is that they often do not implement the newest techniques. For example, the MLP implementation in SPSS Clementine has a basic training algorithm[127] and the search of the weight space uses gradient descent which is not the most efficient or up to date search algorithm. However, the most advanced or most accurate tool may not be the best for a given data mining task. In practice, stability, ease of use, acceptable accuracy, ability to perform the required data mining tasks, and good reporting capabilities, are all more important than using the latest algorithms. Other important considerations for business enterprises are installation, training, and product support. Only data mining suites from a commercial software vendor are likely to offer suitable maintenance or support contracts, and these are of critical importance in an enterprise setting. Data mining suites are easier to use than single task implementations of individual computational knowledge discovery techniques, or Netlab type toolboxes. However, users still require significant knowledge of data analysis, statistics, and databases.

5.2.3 Emerging Technology Trends

Data mining suites are a new class of software application, and represent the second generation of data mining systems. The leading vendors of data mining suites are of two types. Namely, developers of statistical software who have produced data mining suites as companions to their statistics packages, and DBMS vendors who see data mining as a natural extension of their database products. These origins tend to be reflected in their respective products. The data mining suites offered by statistical software vendors[128] are now maturing products featuring refined GUIs, improved performance, and better integration with DBMS and client-server systems. They are beginning to support full KDD processes, and compatibility with CRISP-DM, the SAS Institute's proprietary SEMMA, and other standards. The leading database vendors[129] are currently working on enhancements to run data mining models directly within their

[127] Backpropagation with momentum.
[128] For example SAS Enterprise Miner, SPSS Clementine, and STATISTICA Dataminer.
[129] e.g. IBM, Microsoft, and Oracle.

DBMS. Until very recently vendors have developed data mining software as a general problem-solving tool, which made it difficult for businesses to see how it was relevant to their specific needs. Data mining suites, for example, are horizontal applications offering a collection of tools for different tasks, with limited capacity for customisation. The need however, is for dedicated vertical applications that wrap a complete specialised KDD process in a domain-specific solution, to solve clearly defined business problems. The current trend of placing data mining technology within specific business contexts is customer-driven, and promises to deliver third generation data mining solutions meeting this need. This thesis is a contribution to developing such dedicated vertical solutions, in this case for the business-specific problem of analysing options market data, pricing and hedging options, and forecasting asset prices. There are two types of functionality in particular, not supported by the current generation of generic data mining suites, which third generation vertical solutions for this domain need to address. Namely, statistical analysis within the tool, and derivatives of regression functions.

5.2.4 Statistical Analysis Within the Tool

A statistically principled and coherent KDD and data mining process requires a systematic model search in the Data Mining Phase. An adequate model search must use appropriate statistical tests. In section 2.4.5, the applicability of various statistical and diagnostic tests to a data mining model search suited to options market applications was discussed. Unfortunately, in currently available data mining suites, the tests considered are rarely made available within the package. Data mining suites produced by vendors of statistics software provide connectivity to the vendor's statistics packages, and it is clearly intended that these should be used for any desired statistical testing. This approach allows some basic tests to be performed. However, it is unsuitable for applications to options market data for a number of reasons; General purpose statistics packages do not normally incorporate the full range of statistical and diagnostic tests required for financial work, and may have file size limitations. A dedicated econometrics package is a more appropriate tool. However, even where a separate package incorporates all the required tests there will still be problems. For example, it is not possible to substitute the residuals from a neural net or similar non-parametric regression technique into the standard diagnostic tests for OLS or parametric NLLS regressions. This is because for finite samples, the residuals are not linear functions of

the unknown true errors[130]. Hence, the significance of the test statistic values will be altered. Moreover, a number of important tests including LM tests, require auxiliary regressions to calculate test statistics, and these are more complicated for non-linear regression. It is much easier and more computationally efficient, to implement auxiliary regressions in the same package as the original regression. Therefore, an important specification requirement for third generation dedicated vertical data mining solutions for application to options market databases, is that all necessary statistical and diagnostic tests should be fully integrated, and implemented within the software tool. This would also eliminate the need to export data to separate statistical tools.

5.2.5 Derivatives of Non-parametric Regression Functions

The central role of partial derivatives of option pricing functions with respect to their input parameters, for pricing and hedging options, has been noted earlier in this thesis. It is well known that the first partial derivative of the pricing function with respect to the price of the underlying asset, is the ratio 'Delta', essential for hedging options against movements in the price of the underlying asset. The second partial derivative of the pricing function with respect to the price of the underlying asset is the ratio 'Gamma', used to hedge against the residual risk of changes in 'Delta'. Vega, Rho, and Theta, are the first partial derivatives with respect to volatility, risk free interest rate, and time to maturity, respectively, and are used to hedge against changes in those variables. Finally, the second partial derivative of the option pricing function with respect to the exercise price gives the RND, as discussed in section 4.3.1. Therefore, where the option pricing function is a neural net or other computational knowledge discovery technique for regression, access to the values of the first and second partial derivatives with respect to each input, in addition to the values of the function itself, is essential. Numerical differentiation can be used to compute these values. However, an analytical derivation is preferable and is more accurate in practice. Often the necessary computations are done internally, for sensitivity analysis or pruning purposes. All that is then required is to display results for each observation, which adds negligible computational cost. This functionality should be added as a menu option, allowing the user to select the relevant input variable and degree of differentiation. Surprisingly, this capability is seldom if ever, provided in current software implementations of computational knowledge discovery techniques for non-parametric regression.

[130] Pagan and Ullah (1999) p.208.

5.3 Conclusions

The main objectives of this thesis as outlined in Chapter 1 section 1.1.4 and specified in Appendix H, have been successfully accomplished in that I have developed and presented a complete formal computational framework for the application of computational knowledge discovery techniques to options market databases. The framework, which is described in Chapter 2, supports a full KDD process suitable for third generation dedicated vertical solutions for this domain, and is compliant with the CRISP-DM industry standard. It incorporates a number of practical procedures, methods, and algorithms, designed to reliably estimate the statistical significance and confidence which can be placed in predictions and models. These include; A new robust method for obtaining confidence and prediction intervals for computational knowledge discovery techniques for regression, allowing model predictions to be assessed on a pointwise basis. A related method for estimating the prediction risk performance criterion where target values are unknown, which does not require computationally costly cross-validation. A data mining model search methodology which decomposes the model search task into sequential searches of first, the input space, and then, the space of potential architectures, allowing the use of separate strategies optimised for each search. A shrinkage algorithm for searching the input space, based on the general-to-specific approach, which is robust to the diagnostic problems of regression encountered with options market data. And a growth algorithm for searching the space of potential architectures, designed to avoid the estimation of unidentified models, which uses prediction risk as a selection criterion. These procedures, methods, and algorithms, are contributions to the fields of computational knowledge discovery techniques, KDD and data mining, and model selection, and are relevant to many different domains. In particular, the method for estimating prediction intervals which is presented in Chapter 3, and was extensively tested, allows a fuller evaluation of the uncertainty of model predictions. It permits straightforward uncertainty calculations for large data sets, where ensemble techniques such as bootstrapping are too computationally costly.

The case study examples in Chapter 4, demonstrate that useful, significant, and interesting insights and knowledge can be obtained, when computational knowledge discovery techniques are applied to options market databases using the computational framework, its components, and an appropriate choice of statistical and diagnostic tests.

For finance, the following findings are new contributions; Confirmation, using non-parametric regression techniques, that widely used option pricing models omit significant variables. The finding that transaction costs for the underlying asset are significant for option prices, and that call options should be hedged (priced) using the ask price of the underlying asset, and put options should be hedged (priced) using the bid price, when using the BS option pricing model. And results suggesting that in practice, RNDs from both European and American put options on an asset, can be used interchangeably to obtain unbiased estimates of the future price of that asset on maturity of the options. The use of neural nets to recover RNDs from option prices, as described in section 4.4.2 of the thesis, is a contribution to the field of non-parametric methods of density estimation. The literature documenting the application of computational knowledge discovery techniques to option pricing, has been incomplete in that few results reporting statistical significance or confidence levels, or based on formal hypothesis tests, have been presented. In addition, systematic data mining model search methodologies, and KDD processes, have not been used. The results in this thesis address this omission.

The work in this thesis towards the original goals is substantially complete. However, there are several clear directions for future research. One obvious extension is to develop the data mining model search methodology to include classification, clustering, and rule induction techniques. These techniques are relevant where trading strategies based on regression models must be implemented. The GeTS Model Search Algorithm for Input Space Search given in section 2.4.5, is based on the use of F and t-tests for variable deletion. Use of Wald or LM tests would be more efficient. The former requires only the unrestricted model to be estimated, and the latter only the restricted model, whereas F and t-tests require estimation of both. Thus, a model search would require estimation of fewer models overall. However, to use these tests, methods of adjusting the calculated value of test statistics to allow for non-normal residuals, as well as practical computation procedures, need to be found.

A central problem affecting all available computational knowledge discovery techniques for non-parametric regression, is that there is no conclusive way of choosing the optimal architecture, or degrees of freedom, which should be used. A large literature on this topic exists, but no single procedure has become dominant. Moreover,

the algorithms that are offered mostly rely on heuristic rather than statistical approaches. A statistically based algorithm, using the prediction risk performance criterion was evaluated in section 4.2.3 of this thesis, and found to be of limited capability. This problem is related to the tendency for non-parametric regression techniques to produce models based on a local minimum of the error function, when only the global minimum possesses the required properties. A major theoretical advance in computer science that would provide a solution in most cases, is an efficient and reliable algorithm to locate global minima. The method for estimating prediction intervals described in Chapter 3, should also benefit from better search algorithms. A more effective training algorithm, is a further possibility for improvement in that context. Finally, the results of the statistical tests suggest that the method for estimating RNDs performs well, and offers advantages over the currently established standard methods. However, comprehensive comparative tests, are required to arrive at a definitive determination.

REFERENCES

Ait-Sahalia, Y., and Lo, A. W. 1998. *"Nonparametric Estimation of State Price Densities Implicit in Financial Asset Prices"*, Journal of Finance, Vol. LIII No2, pp 499-547.

Ait-Sahalia, Y., and Lo, A. W. 2000. "*Nonparametric Risk Management and Implied Risk Aversion*", Journal of Econometrics 94, pp 9-51.

Ait-Sahalia, Y., Wang, Y., Yared, F. 2001. *"Do Option Markets Correctly Price the Probabilities of Movement of the Underlying Asset?"* Journal of Econometrics, 102, pp 67-110.

Amilon, H. 2001. "*A Neural Network Versus Black-Scholes: A Comparison of Pricing and Hedging Performances"* Working Paper Department of Economics, Lund University, Lund, Sweden.

Anagnou, I., Bedendo, M., Hodges, S., and Tompkins, R. 2002. "*The Relation Between Implied and Realised Probability Density Functions*", working paper, Financial Options Research Centre, University of Warwick, Coventry, CV4 7AL.

Anders U., Korn O., Schmitt C. 1996. *"Improving the Pricing of Options - A Neural Network Approach"*, ZEW Discussion Paper, pp 96-04.

Anders, U. 1996. *"Statistical Model Building for Neural Networks*", Proceedings of the 6th International AFIR Colloquium, Nürnberg, Germany, October 1-3, 1996, pp 963-979.

Andersson, M., and Lomakka, M. 2003. *"Evaluating Implied RND's by Some New Confidence Interval Estimation Techniques"*, Sveriges Riksbank Working Paper Series, No. 146.

Bahra, Bhupinder. 1997. *"Implied Risk-Neutral Probability Density Functions From Option Prices: Theory and Applications"*, Bank of England, Threadneedle Street, London, EC2R 8AH.

Bakshi, G., Cao, C., and Chen, Z. 1997. *"Empirical performance of alternative option-pricing models"*, The Journal of Finance, Vol.52, No.5, pp 2003-2049.

Bakshi, G., Cao, C., and Chen, Z. 2000. *"Pricing and hedging long-term options"*, Journal of Econometrics. 94, pp 277-318

Benell, J. and Sutcliffe, C. 2000.*"Black-Scholes Versus Artificial Neural Networks in Pricing FTSE 100 Options", Discussion Paper 00-156*, School of Management, Southampton University.

Bentley, C. 2002. *'PRINCE2: A Practical Handbook'*, Butterworth-Heinemann, 2 edn., ISBN: 0 7506 53302.

Bishop, C. M. 1994.*"Mixture Density Networks"*, Technical report NCRG/4288, Neural Computing Research Group, Aston University.

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford.

Bishop, C. M., Qazaz, C. S. 1995. *"Bayesian Inference of Noise Levels in Regression"*, Technical Report, Neural Computing Research Group, Aston University.

Bliss, R., and Panigirtzoglou, N. 2002. *"Testing the Stability of Implied Probability Density Functions"*, Journal of Banking and Finance 26, pp 381-422.

Boyle, P. P., and Vorst, T. 1992. *'Option Replication in Discrete Time with Transaction Costs'*, The Journal of Finance 47, pp 271–293.

Breeden, D. T., and Litzenberger, R. H. 1978. *"Prices of State-contingent Claims Implicit in option Prices"*, Journal of Business 4, pp 621-651.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R. 2000. *"CRISP-DM 1.0"*, CRISP-DM Consortium.

Chidambaran, K. N., Jevons-Lee, C. H., and Trigueros, J. R. 1999. *"An Adaptive Evolutionary Approach to Option Pricing via Genetic Programming"*, Proceedings of the 6th Conference on Computational Finance (formerly Neural Networks in the Capital Markets), Leonard N. Stern School of Business, New York University, January 6-8, 1999.

Cox, J. C., and Ross, S. A. 1976. "*The Valuation of Options for Alternative Stochastic Processes*", Journal of Financial Economics 3, pp 145-166.

Dacorogna, M. M., Gencay, R., Muller, U., Olsen, R. B., Pictet, O. V. 2001. *An Introduction to high Frequency Finance*, Academic Press.

Davis, M. H. A., Panas, V. G., and Zariphopoulou, T. 1993. *'European Option Pricing with Transaction Costs'*, SIAM Journal on Control and Optimisation 31, pp 470–493.

Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L., and Hopfield, J. 1987. *"Large Automatic Learning, Rule Extraction, and Generalisation"*, Complex Systems, 1(5), pp 877-922.

Efron, B., Tibshirani, R. J. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York.

Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. 1996. *"From Data Mining to Knowledge Discovery: An Overview"*, in Advances in Knowledge Discovery and Data Mining, eds. Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. AAAI Press/MIT Press, pp 1-34.

Galai, D. 1983. *"A survey of empirical tests of option-pricing models"*, in Option Pricing: Theory and Applications (M. Brenner, ed.), Lexington Books, Lexington, pp.45-81.

Galindo, J. 1999. *"A Framework for Comparative Analysis of Statistical and machine learning Methods: An Application to the Black-Scholes Option Pricing Equation"*, in Computational Finance. (1999) Abu-Mostafa, Y. S., LeBaron, B., Lo, A. W., and Weigend, A. S. (Eds.) Proceedings of the Sixth International Conference on Computational Finance (CF99, New York, January 1999). Cambridge, MA: MIT Press.

Healy, J. V. 1999. "*Volatility Implied in FTSE 100 Index Option Price Series As a Future Volatility Predictor: A New Approach"*, MSc Dissertation, Department of Economics and Finance, Brunel University London.

Healy, J. V. 2000. *"Data Mining, Machine Learning, and KDD: Applications to Option Market Data"*, Discussion Paper, Department of Computing Communication Systems and Mathematics, London Guildhall University.

Healy, J. V., Dixon, M., Read, B. J. and Cai, F. F. 2001. *" A Data-Centric Approach to Understanding the Pricing of Financial Options*", European Physics Journal B: Proceedings of the 3rd International Conference on Applications of Physics to Financial Analysis 2001.

Healy, J. V., Dixon, M., Read, B. J., and Cai, F. F. 2003. *"Confidence Limits for Data Mining Models of Options Prices"*, European Physics Journal A: Proceedings of the 4th International Conference on Applications of Physics to Financial Analysis 2003.

Healy, J. V., Dixon, M., Read, B. J., and Cai, F. F. 2003. *"Confidence in Data Mining Model Predictions: A Financial Engineering Application"*, 29th Annual Conference of the IEEE Industrial Electronics Society, Special Session on Intelligent Systems.

Hendry, D. F. 1995. *Dynamic Econometrics*. Oxford: Oxford University Press.

Hendry, D. F., and Krolzig, H. M. 2002. *"New Developments in Automatic General-to-specific Modelling"*, paper presented at the ESAM02 conference, Brisbane, July 2002.

Hendry, D. F., and Richard, J. F. 1983. *"The Econometric Analysis of Economic Time Series"*, International Statistical Review, 51, pp 63-111.

Henrotte, P. 1993. *'Transaction Costs and Duplication Strategies'*, Working Paper, Stanford University and Groupe HEC.

Herrmann, R., and Narr, A. 1997. "*Neural Networks and the Valuation of Derivatives - Some Insights into the implied Pricing Mechanism of German Stock Index Options*", Working Paper, University of Karlsruhe, Institute for Decision Theory and Management Science, Department of Finance and Banking.

Heskes, T. 1997. *"Practical confidence and prediction intervals"*, in: M.Mozer, M.Jordan and T.Petsche, eds, Advances in Neural Information Processing Systems 9, MIT Press, Cambridge, MA, pp. 176-182.

Hodges, S. D., and Neuberger, A. 1989. *'Optimal Replication of Contingent Claims under Transaction Costs'*, Review of Futures Market 8, pp 222–239.

Hoggard, T., Whalley, A.E. and Wilmott, P. 1994. *'Hedging Option Portfolios in the Presence of Transaction Costs'*, Advances in Futures and Options. 7, pp 21-35.

Holm, S. 1979. *"A Simple Sequentially Rejective Multiple Test Procedure"*, Scandinavian Journal of Statistics,6, pp 65-70.

Hoover, K. D., and Perez, S. J. 1999. *"Data mining reconsidered: Encompassing and the general-to-specific approach to specification search"*. Econometrics Journal 2, pp 1–25.

Hornik, K., Stinchcombe, M., White, H. 1990. "*Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks*", Neural Networks, Vol. 3, pp. 551-560.

Huber, P. J. 1967. *"The behavior of maximum likelihood estimation under nonstandard conditions"*, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, LeCam, L. M. and Neyman, J. editors. University of California Press, pp 221-233.

Hull, J. 2000. *"Options, Futures and Other Derivative Securities"*, 4th. Ed. Prentice Hall International.

Hutchinson, J., Lo, A. W., and Poggio, T. 1994. *"A Non-parametric Approach to Pricing and Hedging Derivative Securities via Learning Networks."* Journal of Finance, 49, No. 3, pp 851-889.

Hwang, J. T. G., and Ding, A. A. 1997. *"Prediction Intervals for Artificial Neural Networks"*, Journal of the American Statistical Association, Vol. 92 No. 438, p 754.

J. P. Morgan/Reuters, 1996. *"Risk Metrics - Technical Document"*, Fourth Edition, New York.

Jackwerth, J. C. 1999. *"Option Implied Risk-Neutral Distributions and Implied Binomial Trees: A Literature Review"*, Journal of Derivatives,Winter 1999, pp 66-82.

Keber, C. 1999. *"Option Valuation with the Genetic Programming Approach."* Proceedings of the 6th Conference on Computational Finance (formerly Neural Networks in the Capital Markets), Leonard N. Stern School of Business, New York University January 6-8, 1999.

Kelly, D. 1994. "*Valuing and Hedging American Options Using Neural Networks*", Working paper, Carnegie Mellon University.

Lajbcygier, P. 1999. *"Literature Review: The problem with modern Parametric Option Pricing"* Journal of Computational Intelligence in Finance Vol.7 No.5 pp 6-23.

Lajbcygier, P., Connor, J., and Tsang, R. 1999. *"A Constrained Hybrid Approach to Option Pricing"*, Proceedings of the 6th Conference on Computational Finance (formerly Neural Networks in the Capital Markets), Leonard N. Stern School of Business, New York University, January 6-8, 1999.

LeBaron, B., and Weigend, A. S. 1994. *"Evaluating Neural Network Predictors by Bootstrapping"*, In: Proceedings of the International Conference on Neural Information Processing (ICONIP'94), Seoul.

LeBaron, B., and Weigend, A. S. 1998. *"A Bootstrap Evaluation of the Effect of Data Splitting on Financial Time Series"*, IEEE Transactions on Neural Networks 9, pp 213-220.

Leland, H. E. 1985. *'Option Pricing and Replication with Transaction Costs'*, The Journal of Finance 40, pp 1283–1301.

Lovell, M. C. 1983. *"Data Mining"*, Review of Economics and Statistics, 65, pp 1-12.

MacBeth, J., and Merville, L. 1979. "*An Empirical Examination of The Black-Scholes Call Option Pricing Model*", Journal of Finance 34, pp 1173-1186.

Maddala, G. S. 1988. *Introduction to Econometrics*, New York, Macmillian.

Maltz, A. M. 1996. *"Using Option Prices to Estimate Ex-Ante Realignment Probabilities in the European Monetary System: the case of sterling-mark"* , Journal of International Money and Finance, 15, pp 717-48.

Malz, A. M. 1997. *"Estimating the probability distribution of the future exchange rate from options prices"*, Journal of Derivatives 5, pp 20-36.

Melick, W. R., and Thomas, C. P. 1998. "*Confidence Intervals and Constant Maturity Series for Probability Measures Extracted from Option Prices*". Paper presented at the conference 'Information Contained in Prices of Financial Assets', Bank of Canada.

Moody, J. 1994. *"Prediction Risk and Architecture Selection for Neural Nets"*, in From Statistics to Neural Networks: Theory and Pattern Recognition Applications, Cherkassky, V., Friedman, J. H., and Wechsler, H. (eds.), NATO ASI Series F, Springer- Verlag 1994, section 3.1.

Moody, J., and Utans, J. 1991. *"Selecting Neural Network Architectures via the Prediction Risk: Application to Corporate Bond Rating Prediction"*, Proceedings of the First International Conference on Artificial Intelligence Applications on Wall Street. IEEE Computer Society Press, Los Alamitos, CA.

Nabney, I. T. 2002. *Netlab: Algorithms for Pattern Recognition*, Springer Verlag, London.

Neuhaus, H. 1995. "*The Information Content of Derivatives for Monetary Policy*", Discussion Paper 3/95, Economic Research Group of the Deutsche Bundesbank.

Nix, D. A., Weigend, A. S. 1995. *"Learning Local Error Bars for Non-Linear Regression"*, In: Proceedings of NIPS 7, pp 489-496.

Pagan, A., Ullah, A. 1999. *Nonparametric Econometrics*, Cambridge University Press.

Rebonato, R. 1999. *Volatility and Correlation*, Wiley Financial Engineering, John Wiley & Sons Ltd. Chicester.

Rebonato, R. 2003. *"Theory and Practice of Model Risk Management"*, Quantitative Research Centre (QUARC) of the Royal Bank of Scotland, Oxford Financial Research Centre – Oxford University.

Ritchey, R. J. 1990. *"Call Option Valuation for Discrete Normal Mixtures"*, Journal of Financial Research 13, pp 285-296.

Rubinstein, M. 1985. *"Nonparametric Tests of Alternative Option Pricing Models Using all Reported Trades and Quotes on the 30 Most Active CBOE Option Classes from August 23, 1976 through August 31 1978"*, Journal of Finance 40, No.2, pp 455-480.

Satchwell, C. 1994. *"Neural Networks for Stochastic Problems: More than One Outcome for the Input Space"*, In: NCAF Conference, Aston University, September 1994.

Schittenkopf, C., and Dorffner, G. 2000. *"Risk Neutral Density Extraction From Option Prices: Improved Pricing With Mixture Density Networks"*, Working Paper, Austrian Research Institute for Artificial Intelligence, Schottengasse 3, 1010 Vienna, Austria.

Seber, G. A. F., and Wild, C. J. 1989. *Nonlinear Regression*, John Wiley & Sons, New York.

Shimko, D. 1993 *"Bounds of Probability"*, Risk, Vol.6, No.4, pp 33-37.

Sidak, Z. 1967. *"Rectangular Confidence Regions for the Means of Multivariate Normal Distributions"*, Journal of the American Statistical Association, 62, pp 626-633.

Stapleton, J. 1997. *Dynamic Systems Development: The Method in Practice*, Addison-Wesley, Longman, Harlow, England, ISBN 0-201-17889-3.

Terasvirta T., Lin C. F., Granger C. W. 1993. *"Power of the Neural Network Linearity Test"*, Journal of Time Series Analysis, 14(2), pp 209-220.

Thomas, R. L. 1997. *Modern Econometrics*, Addison Wesley.

Tibshirani, R. 1996. *"A comparison of Some Error Estimates for Neural Network Models"*, Neural Computation, 8, pp 152-163.

Tikhonov, A. N., and Arsenin, V. L. 1977. *Solutions of Ill-Posed Problems*, Scripta Series in Mathematics, Halstead Press, Winston, New York.

Titterington, D. M., Smith, A. F. M., and Makov, U. E. 1985. *Statistical Analysis of Finite Mixture Distributions*, John Wiley, New York.

Toft, K. 1996. *'On the Mean-Variance Trade-Off in Option Replication with Transaction Costs'*, Journal of Financial and Quantitative Analysis 31, pp 233–263.

Ungar, L. H., De Veaux, R. D., Rosengarten, E. 1995. *"Estimating Prediction Intervals for Artificial Neural Networks"*, Department of Computer and information Science, University of Pennsylvania, Philadelphia, PA 19104.

White H. 1989. *"An Additional Hidden Unit Test for Neglected Non-linearity in Multilayer Feedforward Networks"*, Proceedings of the International Joint Conference on Neural Networks, Washington, DC. San Diego, SOS Printing, 11, pp 451-455.

White, A. J., Hatfield, G. B., and Dorsey, R. E. 1998. *"A Genetic Algorithm Approach to Pricing Options With Futures-style Margining"*, Working Paper, Murray State University, MS KY 42071.

White, H. 1982. *"Maximum likelihood estimation of misspecified models"*, Econometrica, 50, pp 1-25.

Wu, C. J. F. 1986. *"Jacknife, Bootstrap, and other Resampling Methods in Regression Analysis"*, Annals of Statistics 14, pp 1261-1295.

Glossary of Acronyms and Technical Terms

American Option — An option which can be *exercised* at any time prior to expiration.

Arbitrage — The practice of seeking a profit from situations where the same good is offered at different prices, either in different markets or in the same market at different times.

At-the-Money — An option is said to be 'at-the-money' when the price of the underlying asset equals its strike (exercise) price. In this condition it will have an intrinsic value exercised immediately.

Bandwidth — Smoothing parameter. A term synonymous with bin width for a histogram, window width for kernel methods, number of hidden layer nodes for a neural net, the value of K, for KNN etc. Effectively, the permitted degrees of freedom.

Black-Scholes Model — A mathematical model (equation) for pricing *European options* on a non-dividend-paying underlying asset.

Brownian Motion — A *Markov* stochastic process which is the limit of a random walk as the time interval $\Delta t \rightarrow 0$. It has zero drift and unit volatility, i.e. $dz \sim N(0,1)$ where z is a variable subject to the process. It is a *martingale* and is not Newtonian differentiable. It is sometimes referred to as a *Weiner* process.

Call Option — An option to buy (i.e. 'call for') an underlying asset.

Change of Measure — A transformation of a probability distribution which changes the mean, but not the higher moments. It can be shown to correspond to changing the set of assumed risk preferences. Useful in option pricing since it is often easier to work with *risk neutral probabilities*.

CRISP-DM — Cross Industry Standard Process for Data Mining

Data Mining — Defn. A step in the KDD process consisting of particular data mining algorithms that, under some acceptable computational efficiency limitations, produce a particular enumeration of patterns

DBMS — Data Base Management System

Delta — of an option. A *hedge ratio*. The first order partial

	derivative of the option pricing function with respect to the price of the underlying asset.
European Option	An option which can be exercised only at expiration.
Exercise	verb. To make use of the contractual right conferred by an option to buy or sell the underlying asset.
Exercise Price	See Strike Price.
Expiration	The date and time when an option contract expires, also called maturity.
Functional	A function that has a domain that is a set of functions and a range belonging to another set of functions. For example, the differential operator d/dx is a functional of differentiable functions $f(x)$. The range of the functional may be a set of numbers. An example of this is a definite integral of $f(x)$ with respect to x.
Futures contract	A contract between two parties for the purchase of an asset at a future date, the price being agreed when the contract is struck.
GANN	Genetically Adaptive Neural Net
Geometric Brownian Motion	The most popular *stochastic process* for modelling stock prices. The change in the log of a variable z subject to geometric Brownian motion in a short time interval Δt is normally distributed with mean and variance both proportional to Δt. The process is the exponential of a *Brownian motion* with drift.
GeTS	General to Specific Search
GLS	Generalised Least Squares
Hedge	verb. To attempt to offset potential losses on a contract by means of an equal and opposite transaction using options (or some other instrument).
Hedge Ratio	of an option. The partial derivative of the option pricing function with respect to the variable (e.g. the price of the underlying asset) whose movement is to be hedged).
Hessian Matrix	Matrix of 2^{nd} order partial derivatives of a function.

Heteroskedastic — Rigorously, the error terms of an unknown true regression model are said to be heteroskedastic if they have a non-constant variance. The term is often loosely used to describe residuals from an estimated regression model which exhibit non-constant variance.

In-the-Money — An option is said to be 'in the money' if immediate exercise will realise a gross profit.

Intrinsic Value — The profit (if any) realised by immediate exercise of an option.

KDD — Knowledge Discovery in Databases. Defn. The non trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data

KNN — K-Nearest Neighbour. A classification algorithm.

Leverage — The use of debt to finance investment, hence magnifying returns on the non-debt portion of the capital sum invested.

LIBOR — London Inter Bank Offer Rate, interest rate on Eurodollar deposits, often used as a proxy for 'risk-free' rate in UK options markets.

Markov Process — A discrete *stochastic process* where the next value of a variable is conditional on its present value but independent of all previous values.

Martingale — A *stochastic process* with zero drift where the expected value of a variable at any future time, conditional on the past, is its present value.

Martingale Measure — A probability distribution (measure) under which a process is a *martingale.* Often referred to as an Equivalent Martingale Measure.

Measure — A set of probabilities (probability distribution) specifying the likelihood of each of all possible outcomes of a particular event. It can be shown to correspond to a set of risk preferences.

MLP — Multi Layer Perceptron. Feed-forward (i.e. with no feedback loops) neural networks with multiple layers of adaptive weights.

NLLS Non Linear Least Squares

NN Neural Net. A form of regression where the mapping between the dependent variable or *target* and the explanatory or *input variables* is modelled by a linear combination of weighted basis functions. These may be linear, sigmoidal, or step functions. The weights corresponding to the best fit to the data are iteratively estimated during training.

OLS Ordinary Least Squares. Linear regression

Option A contract to buy or sell an asset at an agreed price, at or before some agreed future time.

OTC Option A customised options contract (i.e.'over the counter') designed to meet a specific requirement, for which there is normally no market.

Out of the Money An option is said to be 'out of the money' if immediate exercise will realise a loss.

PPR Prior Pursuit Regression

PR Prediction Risk. The expected value of the mean squared error $E(MSE)$.

Premium The purchase price of an option.

Put Option An option to sell (i.e.'put on the market') an underlying asset.

Random Walk A discrete *Markov process* composed of the sum of a number of independent steps. In a simple symmetric random walk, at each step a variable x increases by 1 with probability 0.5 or decreases by 1 also with probability 0.5.

RBF Radial Basis Function

RDBMS Relational Data Base Management System

Replicating Portfolio of an option. A dynamically adjusted portfolio of bonds and the underlying asset (or futures on the underlying asset) with hedge ratios maintained equal to those of the option being replicated (synthesized). The position required to replicate an option is the reverse of that required to hedge it.

Risk-free rate — An interest rate or return, equal to that obtainable from a portfolio of assets giving a guaranteed return, such as Government Bonds.

RND — The risk-neutral probability distribution of the underlying asset taking the values of adjacent strike prices at expiration. More rigorously, the State Price Density or set of probabilities of the option paying $1 if the underlying asset assumes the value of a particular strike price at expiration and $0 if it assumes any other value, worked out for all possible contemporaneous strike prices and option prices on the asset. It can be converted to the true probability distribution for the value of the asset at expiration by a *change of measure*.

Stochastic — A synonym for random.

Stochastic Process — An equation describing the probabalistic behaviour of a variable whose future value is uncertain. More rigorously, a continuous process that can be decomposed into a *Brownian motion* term and a drift term (i.e. a stochastic component and a deterministic component).

Strike Price — The price at which the option holder may buy or sell the underlying asset, as defined in an option contract. Also known as the *exercise price*.

Target dataset — A dataset or subset of variables or data samples on which KDD operations or data mining are to be performed.

Traded Options — Standardised options contracts available (usually) with expiration dates 1,2 or 3 months in the future and traded on an exchange.

Volatility — The annual standard deviation of daily price changes. A measure of the amount by which a price is expected to change.

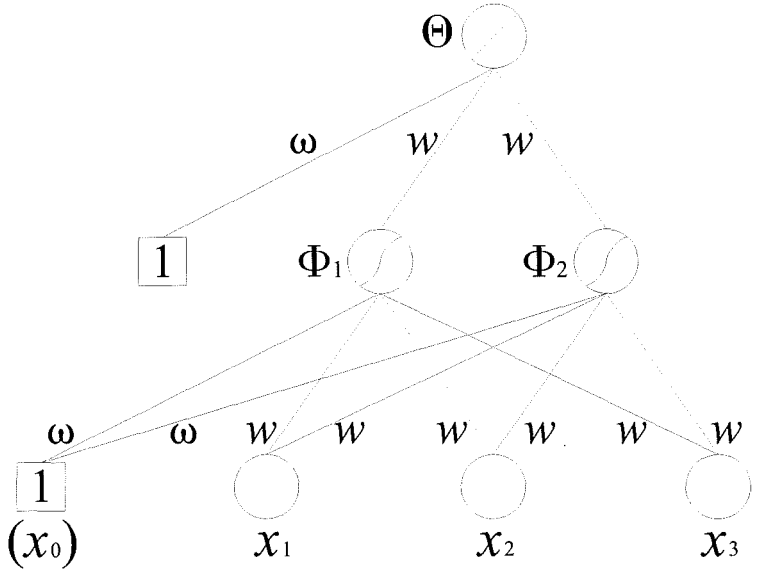
Write — To sell an option in an opening transaction. The writer assumes an obligation to supply (or take delivery of) the underlying asset at the strike price if the option is exercised. In return for assuming the risk inherent in this obligation a premium is received.

APPENDIX A: Neural Networks

This thesis made use of a class of neural network (NN); the multi-layer perceptron (MLP). MLPs are *feed-forward* (i.e. with no feedback loops) networks with multiple layers of *adaptive weights*. They have either threshold or sigmoidal *activation functions*. Where the activation functions are differentiable, MLPs make use of the technique of *error back-propagation* to evaluate the gradients of the error (cost) function with respect to the weights and biases in the network. These gradients are used by an iterative optimisation algorithm, to search the weight space for the global minimum of the error function.

The specific software implementation used in connection with this thesis was a standard MLP using back-propagation, together with a sequential *gradient descent* search algorithm. The hidden unit activation functions were logistic functions, and the output units used linear functions. *Early stopping* was used to terminate training. A single layer of hidden units and two layers of adaptive weights were used. Fig.1 shows the basic structure of such a MLP with, in this case, a single output.

Fig. 1 MLP with a Single Hidden Layer and a Single Output

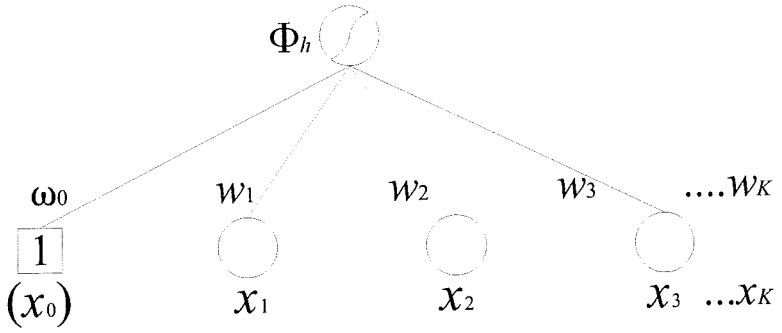


The input nodes are not activation functions, but simply take the values of the input variables x_i. The input x_0 is a constant which is permanently set to $x_0 = 1$. Three input variables are shown, but in principle there is no restriction on the maximum number. The ω are known as biases, and the w are weights. The weights and biases are randomly initialised with small values, and the set of weights and biases corresponding to the minimum of the error function is determined during network training (fitting). The Φ_h are hidden units, two are shown in Fig.1, but again there is no limit to the number, in principle. The hidden units have a logistic activation function so that

$$\Phi_h = f(x) = \frac{1}{1+e^{(-x)}} \qquad \text{(A.1)}$$

Fig.2 shows the structure of a typical hidden unit. The output consists of a weighted linear combination of the k input variables with a bias term added, transformed by the application of the logistic activation function.

Fig. 2 Typical Hidden Layer Node



The function corresponding to Fig.2 can be written as

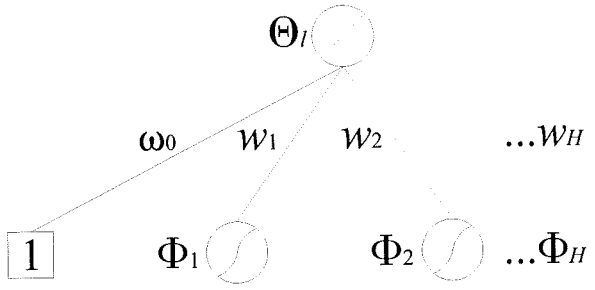
$$\Phi_h(\omega_0 + w_1x_1 + w_2x_2 + w_3x_3 +,...,+w_Kx_K) \equiv \Phi_h\left(\sum_{k=1}^{K} w_{hk}x_k + \omega_h\right) \tag{A.2}$$

The output units have a linear activation function such that

$$\Theta_l = g(x) = x \tag{A.3}$$

The structure of a typical output unit is shown in Fig.3. The output consists of a weighted linear combination of the h hidden units in the preceding hidden layer with a bias term added, transformed by the application of the linear activation function.

Fig. 3 Typical Output Node

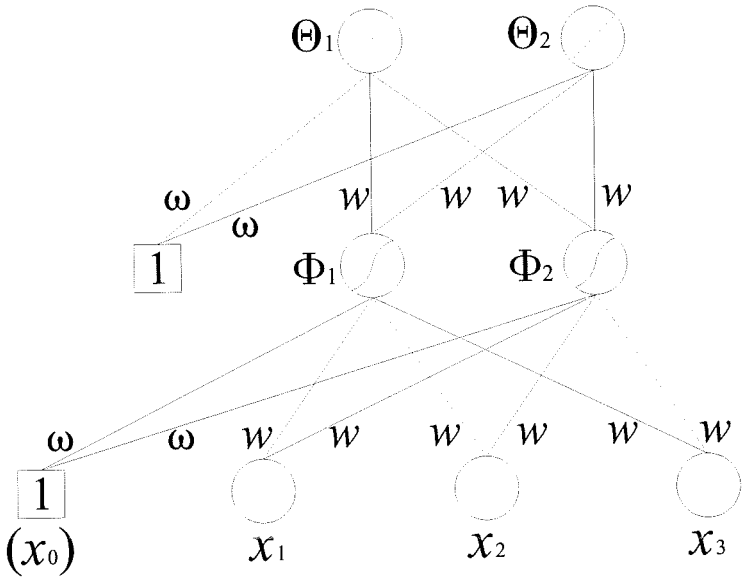


The function corresponding to Fig.3 can be written as

$$\Theta_l(\omega_0 + w_1\Phi_1 + w_2\Phi_2 +,...,+w_H\Phi_H) \equiv \Theta_l\left(\sum_{h=1}^{H} w_{lh}\Phi_h + \omega_l\right) \tag{A.4}$$

MLPs can have multiple outputs. Fig.4 shows a MLP with two outputs. However, regardless of whether there is a single, or multiple outputs, the structure of the hidden layer nodes and output nodes is the same.

Fig. 4 MLP with Multiple Output Nodes

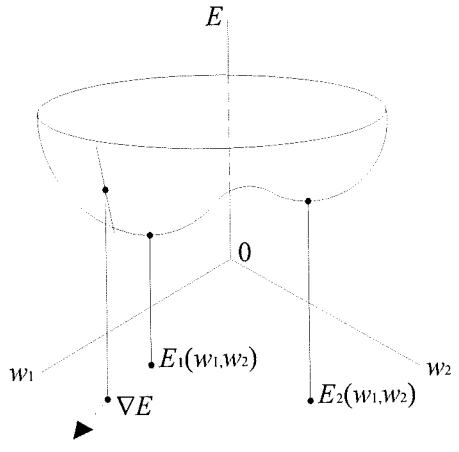


The function corresponding to the network shown in Fig.4 is given by.

$$\hat{\mu}_y(\boldsymbol{x};\hat{\boldsymbol{\Omega}}) = \boldsymbol{\Theta}\left(\sum_{h=1}^{H} w_{lh}\Phi_h\left(\sum_{k=1}^{K} w_{hk}x_k + \omega_h\right) + \omega_l\right) \qquad \text{(A.5)}$$

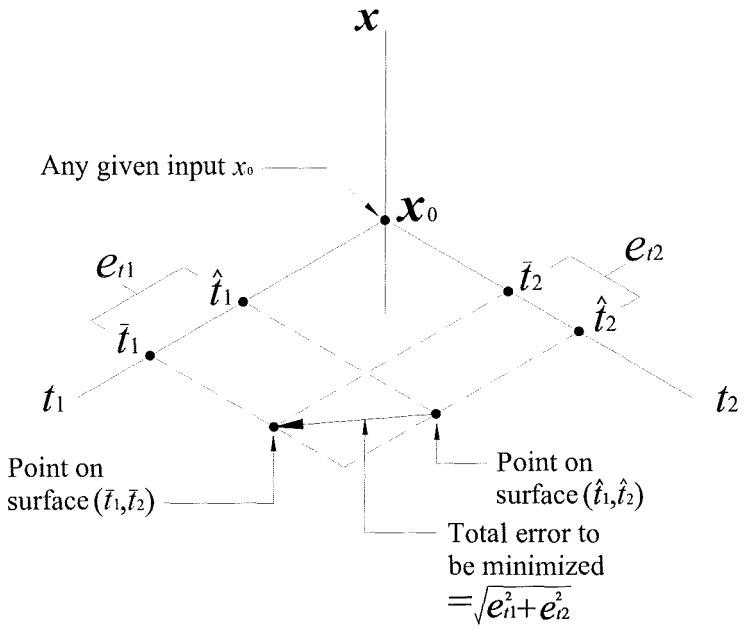
In equation (A.5), $\boldsymbol{\Theta}$ is a vector of separate outputs, with 2 elements in the case of the network in Fig.4. The function (A.2) for a typical hidden layer node is nested within the function (A.4) for a typical output node shown in Fig.3, to give an expression for the complete network. The network shown in Fig.1 is a special case of the network in Fig.4, with l=1. To understand how a network can fit more than one target variable it is necessary to consider the nature of the error function. Geometrically, the error function for a neural net can be thought of as a multidimensional surface located above a multidimensional weight space. Fig.5 is a diagrammatic representation of an error surface. For illustrative convenience, the surface shown is restricted to three dimensions, and the weight space to two dimensions. In practice the dimensionality of both can be much larger.

Fig.5 Error Surface in Error/Weight Space



In Fig.5 the local gradient of the error surface at any point is given by the vector ∇E. The point $E_1(w_1,w_2)$ on the error surface corresponds to the global minimum of the error surface, and the point $E_2(w_1,w_2)$ corresponds to a local minimum. Adding more outputs to a network increases the dimensionality of the weight space and the complexity of the error function. Fig.6 illustrates the error for a network using least square error functions, with two outputs to fit two independently distributed target variables.

Fig.6 Two targets: Individual and Total Errors



In Fig.6 $\boldsymbol{x}_0$ is any arbitrary scalar or vector valued input to the network. The two corresponding actual output values are $\bar{t}_1$, and $\bar{t}_2$. The network estimates of these values are $\hat{t}_1$ and $\hat{t}_2$ respectively. The error in estimating $\bar{t}_1$ is $e_{t_1} = (\bar{t}_1 - \hat{t}_1)$, and for $\bar{t}_2$ the estimation error is $e_{t_2} = (\bar{t}_2 - \hat{t}_2)$, giving a combined error of $\sqrt{e^2_{t_1} + e^2_{t_2}}$. The squared errors are $e^2_{t_1} = (\bar{t}_1 - \hat{t}_1)^2$ and $e^2_{t_2} = (\bar{t}_2 - \hat{t}_2)^2$ respectively. In the case of networks with multiple outputs, the sum of squared error function therefore takes the form[1];

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{m}(\bar{t}_l - \hat{t}_l)^2 \qquad \text{(A.6)}$$

in equation (A.6) n is the number of observations in the data set, and m is the total number of outputs for the network. During the training process the iterative search algorithm will seek the set of weights corresponding to the minimum of this error function. For a network with two outputs, as shown in Fig.4 and discussed in relation to Fig.6, $m = 2$.

[1] See Bishop C. M., 1995. *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford. p196.

APPENDIX B: Delta and Sandwich Methods

B.1 The Delta Method

Consider a sample $\boldsymbol{S}$, $\{(\boldsymbol{x}_1,y_1), (\boldsymbol{x}_2,y_2), \ldots, (\boldsymbol{x}_n,y_n) \in \boldsymbol{S}\}$ where y represents scalar targets and $\boldsymbol{x}$ a vector of k inputs. Suppose the true relationship between the targets y and the input vectors $\boldsymbol{x}$ takes the form:

$$y_i = f(\boldsymbol{x}_i;\boldsymbol{\beta}) + \varepsilon_i \quad (i=1,\ldots,n) \tag{B.1}$$

The data is modelled by the regression equation (3.2) where the right hand side of (3.2) is a MLP. Thus the vector of parameters $\boldsymbol{\beta}$ is replaced by the set of weights and biases $\boldsymbol{\Omega}$, and;

$$\hat{\mu}_y(\boldsymbol{x}_i;\hat{\boldsymbol{\Omega}}) = f(\boldsymbol{x}_i;\hat{\boldsymbol{\Omega}}) + e_i \quad (i=1,\ldots,n) \tag{B.2}$$

If n is large enough so that $\hat{\boldsymbol{\Omega}} \approx \boldsymbol{\Omega}$, a local linear approximation of the network about $\boldsymbol{x}_0 = \boldsymbol{x}$, where $\boldsymbol{x}_0$ is a particular value of $\boldsymbol{x}$, can be obtained and the procedures for multivariate linear regression applied. From a first-order Taylor series,

$$\hat{\mu}_y(\boldsymbol{x}_0;\hat{\boldsymbol{\Omega}}) = f(\boldsymbol{x}_0;\hat{\boldsymbol{\Omega}}) \approx f(\boldsymbol{x}_0;\boldsymbol{\Omega}) + \boldsymbol{g}_0^T \Delta\hat{\boldsymbol{\Omega}} \tag{B.3}$$

where $\Delta\hat{\boldsymbol{\Omega}} = (\hat{\boldsymbol{\Omega}} - \boldsymbol{\Omega})$, and $\boldsymbol{g}_0 = df(\boldsymbol{x}_0;\boldsymbol{\Omega})/d\boldsymbol{\Omega}$.

Substituting this approximation into the least-squares cost function gives the following expression for the sum of squared residuals;

$$\begin{aligned} SSR &\approx \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i;\boldsymbol{\Omega}) - \boldsymbol{g}_i^T \Delta\hat{\boldsymbol{\Omega}})^2 \\ &\approx \sum_{i=1}^{n} (e_i - \boldsymbol{g}_i^T \Delta\hat{\boldsymbol{\Omega}})^2 \end{aligned} \tag{B.4}$$

Rewriting (B.4) in matrix notation gives,

$$SSR \approx (\boldsymbol{e} - \boldsymbol{G}\Delta\hat{\boldsymbol{\Omega}})^T (\boldsymbol{e} - \boldsymbol{G}\Delta\hat{\boldsymbol{\Omega}}) \tag{B.5}$$

where $\boldsymbol{G}$ is an $n \times k$ matrix whose i^{th} row is the vector $\boldsymbol{g}_i^T$ and $\boldsymbol{e}$ is an $n \times 1$ vector of errors. Setting the derivatives of (B.5) with respect to $\hat{\boldsymbol{\Omega}}$ equal to zero and solving the resulting equations to minimise the *SSR* gives;

$$\Delta\hat{\boldsymbol{\Omega}} \approx (\boldsymbol{G}^T\boldsymbol{G})^{-1}\boldsymbol{G}^T\boldsymbol{e} \tag{B.6}$$

Different samples will generate different weight vectors according to (B.6). The variance-covariance matrix of the weights is (the derivation is not given here),

$$\begin{aligned} \widehat{Var}(\Delta\hat{\boldsymbol{\Omega}}) &\approx s_y^2(\boldsymbol{G}^T\boldsymbol{G})^{-1} \\ &\approx \boldsymbol{H}^{-1} \end{aligned} \tag{B.7}$$

where $\boldsymbol{H}^{-1}$ is the outer product approximation to the Hessian matrix of second order partial derivatives of the cost function (B.5) with respect to each of the elements of the weight matrix $\hat{\boldsymbol{\Omega}}$. From (B.3) and (B.7) the standard error of the regression function can now be defined as;

$$\begin{aligned}\widehat{SE}(f(\boldsymbol{x}_i;\boldsymbol{\Omega})) &\approx \left(\sqrt{\boldsymbol{g}_i^T(\boldsymbol{G}^T\boldsymbol{G})^{-1}\boldsymbol{g}_i s_y^2}\right) \\ &\approx \left(\sqrt{\boldsymbol{g}_i^T\boldsymbol{H}^{-1}\boldsymbol{g}_i}\right)\end{aligned} \tag{B.8}$$

Using equation (A.8), a (1-α) 100% confidence interval for $\mu_y(\boldsymbol{x}_0)$ is given by,

$$\hat{\mu}_y(\boldsymbol{x}_0^T;\hat{\boldsymbol{\Omega}}) \pm t_{(\alpha/2)(n-k-1)}\left(\sqrt{\boldsymbol{g}_0^T(\boldsymbol{G}^T\boldsymbol{G})^{-1}\boldsymbol{g}_0 s_y^2}\right) \tag{B.9}$$

and a (1-α) 100% prediction interval is;

$$\hat{\mu}_y(\boldsymbol{x}_0^T;\hat{\boldsymbol{\Omega}}) \pm t_{(\alpha/2)(n-k-1)}\left(\sqrt{(1+\boldsymbol{g}_0^T(\boldsymbol{G}^T\boldsymbol{G})^{-1}\boldsymbol{g}_0)s_y^2}\right) \tag{B.10}$$

Equations (B.9) and (B.10) have the same form as equations (3.7) and (3.8) in Chapter 3, section 3.2.1, with vector $\boldsymbol{g}$ substituted for $\boldsymbol{x}$ and matrix $\boldsymbol{G}$ substituted for $\boldsymbol{X}$. The s_y^2 term is SSR/(n-k-1), the MSR obtained using (B.5) in the numerator. Note that k here is the number of effective weights and biases.

If regularisation as described by Bishop (1995) rather than early stopping is used to prevent over fitting, the standard error given in equation (A.8) must be replaced by,

$$\widehat{SE}(f(\boldsymbol{x}_i;\boldsymbol{\Omega})) \approx \left(\sqrt{\boldsymbol{g}_i^T(\boldsymbol{H}^{-1}+2\lambda)\boldsymbol{g}_i}\right) \tag{B.11}$$

where the cost function to be minimised is,

$$SSR + \lambda\sum_{i=1}^{n} w_i^2 \tag{B.12}$$

and the λ term in (B.12) is a penalty term used to induce weight decay. The w_i are the weights from equation (3.11).

The value of k, the degrees of freedom to use in equations (B.9) and (B.10) is somewhat problematic. When neural net training is stopped before convergence, by regularisation or early stopping, some weights and biases may be ineffective. An architecture selection algorithm should be applied, to ensure there are no redundant hidden layer nodes, thereby minimising or eliminating redundant weights. Assuming there are no redundant weights or hidden units in the network, $k = 1 + H\,(q + 2)$, where q is the number of input variables, and H the number of hidden layer nodes. Otherwise, this is

an upper bound. In either case, for large data sets (e.g. 7000 + observations), provided $k \leq 50$ use of the above should have minimal effect on estimation accuracy.

B.2 The Sandwich Method

An important assumption underlying the OLS and delta method estimators of standard error concerns the variance of the noise ε associated with the true regression. This is assumed $\varepsilon \sim N(0,\sigma^2)$ with constant variance σ^2. The presence of heteroskedasticity will result in estimated standard errors that are biased. The sandwich estimator[1] provides a method of dealing with this problem. The sandwich estimator is obtained by replacing the variance-covariance matrix of weights given by equation (B.7) with;

$$\widehat{Var}_{Sand}(\Delta\hat{\Omega}) \approx \frac{n\left[(\boldsymbol{G}^T\boldsymbol{G})^{-1}\boldsymbol{G}^T(\sum_{i=1}^{n}\boldsymbol{g}_i\boldsymbol{g}_i^T e_i^2)\boldsymbol{G}(\boldsymbol{G}^T\boldsymbol{G})^{-1}\right]}{(n-k)} \tag{B.13}$$

Substituting (B.13) into (B.8) in place of (B.7) gives;

$$\widehat{SE}_{Sand}(f(\boldsymbol{x}_i;\boldsymbol{\Omega})) \approx \left(\sqrt{\boldsymbol{g}_i^T\left(\widehat{Var}_{Sand}(\Delta\hat{\Omega})\right)\boldsymbol{g}_i}\right) \tag{B.14}$$

Equation (B.13) yields asymptotically consistent variance-covariance matrix estimates without making distributional assumptions, even if the assumed model underlying the parameter estimates is incorrect. Because of these desirable properties, the sandwich estimator is also termed the *robust covariance matrix estimator* or the *empirical covariance estimator.*

[1] Huber, P. J. 1967. *"The Behavior of Maximum Likelihood Estimation Under Nonstandard Conditions"*, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, LeCam, L. M. and Neyman, J. editors. University of California Press, pp221-233.
White, H. 1982. *"Maximum Likelihood Estimation of Misspecified Models"*. Econometrica, 50, pp1-25.

APPENDIX C: Bootstrap and Bayesian Methods

C.1 Bootstrap Pairs method

Resampling methods provide an alternative to the delta method for calculating standard errors and statistical intervals for neural nets. The *bootstrap method*[1] is a computer-based technique based on resampling that can provide confidence intervals for any population parameter estimate. In the context of regression, two forms of bootstrap are possible. The first of these is the *bootstrap pairs* method. Consider a sample $\boldsymbol{S}$, $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2,y_2), \ldots, (\boldsymbol{x}_n, y_n) \in \boldsymbol{S}\}$, where y represents scalar targets and $\boldsymbol{x}$ a vector of k inputs. A *bootstrap sample* is a sample $\boldsymbol{S}^{Boot}$, $\{(\boldsymbol{x}_i,y_i) \in \boldsymbol{S}^{Boot}, i = 1 \text{ to } n\}$, consisting of n pairs of $(\boldsymbol{x}_i, y_i)$ drawn randomly (with replacement) from $\boldsymbol{S}$. This means that some $(\boldsymbol{x}_i, y_i)$ may appear more than once in $\boldsymbol{S}^{boot}$ while others may not appear at all. The bootstrap estimate of the standard error of the true regression function, which is a function of the set of inputs $\boldsymbol{X}$, is given by;

$$\widehat{SE}_{Boot}(f(\boldsymbol{X};\boldsymbol{\Omega})) \approx \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left[\hat{\mu}_y(\boldsymbol{X}_b;\hat{\boldsymbol{\Omega}}) - \hat{\mu}_{y,Boot}(\boldsymbol{X})\right]^2} \tag{C.1}$$

In equation (C.1), $\hat{\mu}_y(\boldsymbol{X}_b;\hat{\boldsymbol{\Omega}})$ is the neural net trained on the b^{th} bootstrap sample $\boldsymbol{S}_b^{Boot}$, where there are a total of B bootstrap samples, with typical values $20 < B < 200$. The bootstrap estimate of the mean of the target values $\hat{\mu}_{y,Boot}(\boldsymbol{X})$, termed a *bagged* estimate in the neural net literature, is given by the mean of the ensemble of B networks;

$$\hat{\mu}_{y,Boot}(\boldsymbol{X}) \approx \frac{1}{B}\sum_{b=1}^{B}\hat{\mu}_y(\boldsymbol{X}_b;\hat{\boldsymbol{\Omega}}) \tag{C.2}$$

In equations (C.1) and (C.2), $\boldsymbol{X}$ is an n $\times$ 1 vector of the $\boldsymbol{x}_i$, that is, an $n \times k$ matrix of x values. Using equation (C.1), a (1-α) 100% bootstrap confidence interval for $\mu_y(\boldsymbol{X})$ is given by[2];

$$\hat{\mu}_{y,Boot}(\boldsymbol{X}) \pm t_{(\alpha/2)B}\left(\widehat{SE}_{Boot}(f(\boldsymbol{X};\boldsymbol{\Omega}))\right) \tag{C.3}$$

[1] Efron, B., Tibshirani, R.J. 1993. *An Introduction to the Bootstrap*. Chapman and Hall, New York.
[2] Heskes, T. 1997. *"Practical Confidence and Prediction Intervals"*, in: M. Mozer, M. Jordan and T. Petsche, eds., `Advances in Neural Information Processing Systems 9', MIT Press, Cambridge, MA, pp176-182.

C.2 Bootstrap Residuals Method

In the *bootstrap residuals* method the residuals from a neural net $\hat{\mu}_y(\boldsymbol{X};\hat{\boldsymbol{\Omega}})$ trained on a sample $\boldsymbol{S}$, defined as before, are resampled rather than the training sample itself. Suppose $\boldsymbol{E}$, $\{e_1, e_2, \ldots, e_n) \in \boldsymbol{E}\}$ is a set of residuals from $\hat{\mu}_y(\boldsymbol{X};\hat{\boldsymbol{\Omega}})$ trained on sample $\boldsymbol{S}$. A *bootstrap residuals sample* is a sample $\boldsymbol{E}^{Boot}$, $\{e_i \in \boldsymbol{E}^{Boot}, i = 1 \text{ to } n\}$ consisting of n samples e_i drawn randomly (with replacement) from $\boldsymbol{E}$. The bootstrap residual estimate of the standard error of the true regression function is given by;

$$\widehat{SE}_{BootR}(f(\boldsymbol{X};\boldsymbol{\Omega})) \approx \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left[\hat{\mu}_y^r(\boldsymbol{X};\hat{\boldsymbol{\Omega}}^b) - \hat{\mu}_{y,BootR}(\boldsymbol{X})\right]^2} \tag{C.4}$$

In equation (C.4), $\hat{\mu}_y^r(\boldsymbol{X};\hat{\boldsymbol{\Omega}}^b)$ is the NN trained on the b^{th} bootstrap residual sample $\boldsymbol{E}_b^{Boot}$, where there are a total of B bootstrap residual samples, with typical values $20 < B < 200$. The target for $\hat{\mu}_y^r(\boldsymbol{X};\hat{\boldsymbol{\Omega}}^b)$ is $(\boldsymbol{E}_b^{Boot} + \hat{\mu}_y(\boldsymbol{X};\hat{\boldsymbol{\Omega}}))$. The bootstrap residual estimate of the mean of the target values, $\hat{\mu}_{y,BootR}(\boldsymbol{X})$, is given by the mean of the ensemble of B networks;

$$\hat{\mu}_{y,BootR}(\boldsymbol{X}) \approx \frac{1}{B}\sum_{b=1}^{B}\hat{\mu}_y^r(\boldsymbol{X};\hat{\boldsymbol{\Omega}}^b) \tag{C.5}$$

Using equation (C.4), a (1-α) 100% bootstrap confidence interval for $\mu_y(\boldsymbol{X})$ is;

$$\hat{\mu}_{y,BootR}(\boldsymbol{X}) \pm t_{(\alpha/2)B}\left(\widehat{SE}_{BootR}(f(\boldsymbol{X};\boldsymbol{\Omega}))\right) \tag{C.6}$$

The bootstrap residuals method has the advantage that the same sample $\boldsymbol{S}$ is the source of the inputs $\boldsymbol{X}$ for all B networks that must be trained. This may be an advantage in some experimental situations. On the other hand, it is model specific and not as robust to over fitting or mis-specification as the bootstrap pairs method.

C.3 Bayesian Approaches

The delta, sandwich, and bootstrap estimators of standard errors are based on the maximum likelihood framework. Bayesian statistics provides a different approach. In classical "frequentist" statistics, inferences about the parameters of a population $\boldsymbol{P}$ are based entirely on sample statistics from sample(s) $\boldsymbol{S}$ drawn randomly from $\boldsymbol{P}$. Bayesian statistics in contrast, takes account of prior beliefs about the population $\boldsymbol{P}$, by basing inferences on a *prior probability distribution* that is combined with a sample $\boldsymbol{S}$ to

produce a *posterior (probability) distribution* $P(\theta|\boldsymbol{S})$, for the parameter of interest θ. Confidence and prediction intervals are defined within the Bayesian Framework.

Let θ be a parameter of the population distribution $\boldsymbol{P}$, and $\boldsymbol{S}$ a random sample drawn from $\boldsymbol{P}$. If θ is viewed as a random variable whose posterior distribution is $P(\theta|\boldsymbol{S})$, then $[\lambda_L(\boldsymbol{S}), \lambda_U(\boldsymbol{S})]$ is a $(1-\alpha)100\%$ *Bayesian confidence interval* for θ if from $P(\theta|\boldsymbol{S})$ there is a $(1-\alpha)100\%$ probability $\theta \in [\lambda_L(\boldsymbol{S}), \lambda_U(\boldsymbol{S})]$. In the Bayesian approach, θ is a random variable and $[\lambda_L(\boldsymbol{S}), \lambda_U(\boldsymbol{S})]$ is fixed given availability of $\boldsymbol{S}$. In the classical approach, it is θ which is fixed and $[\lambda_L(\boldsymbol{S}), \lambda_U(\boldsymbol{S})]$ varies with $\boldsymbol{S}$. If $\boldsymbol{S}$ is a (univariate) random sample drawn from $\boldsymbol{P}$ where $\{(x_1, x_2, \ldots, x_n) \in \boldsymbol{S}, n<p\}$ and $P(x_{n+1}|\boldsymbol{S})$ is the posterior distribution for x_{n+1}, then $[\psi_L(\boldsymbol{S}), \psi_U(\boldsymbol{S})]$ is a $(1-\alpha)100\%$ *Bayesian prediction interval* for x_{n+1} if from $P(x_{n+1}|\boldsymbol{S})$ there is a $(1-\alpha)100\%$ probability $x_{n+1} \in [\psi_L(\boldsymbol{S}), \psi_U(\boldsymbol{S})]$. For regression, maximum likelihood based methods estimate single values for each (unknown) parameter of the true regression. The Bayesian approach, in contrast, expresses the uncertainty regarding the true weight vector $\boldsymbol{\Omega}$ as the posterior probability distribution $P(\boldsymbol{\Omega}|\boldsymbol{S})$ given a sample $\boldsymbol{S}$. Thus,

$$\begin{aligned} P(\mu_y(\boldsymbol{x})|\boldsymbol{S}) &= \int_{\boldsymbol{\Omega}} P(\mu_y(\boldsymbol{x})|\boldsymbol{\Omega})P(\boldsymbol{\Omega}|\boldsymbol{S})d\boldsymbol{\Omega} \\ &\propto \int_{\boldsymbol{\Omega}} P(\mu_y(\boldsymbol{x})|\boldsymbol{\Omega})P(\boldsymbol{S}|\boldsymbol{\Omega})P(\boldsymbol{\Omega})d\boldsymbol{\Omega} \end{aligned} \tag{C.7}$$

where $P(\boldsymbol{\Omega})$ is the prior distribution for the weights. MacKay[3] shows that with approximations, the latter integral can be solved analytically. If the noise is assumed to be $\varepsilon \sim N(0,\sigma^2)$ and the prior $P(\boldsymbol{\Omega})$ is also assumed to be Gaussian, then a Gaussian posterior distribution $P(\mu_y(\boldsymbol{x})|\boldsymbol{\Omega}_{MP})$ can be derived where;

$$\hat{E}[\mu_y(\boldsymbol{x})] = \hat{\mu}_y(\boldsymbol{x};\boldsymbol{\Omega}_{MP}) \tag{C.8}$$

In (C.8), $\boldsymbol{\Omega}_{MP}$ is $\boldsymbol{\Omega}$ at the maximum of the posterior probability distribution $P(\boldsymbol{\Omega}|\boldsymbol{S})$. The variance of $P(\mu_y(\boldsymbol{x})|\boldsymbol{\Omega}_{MP})$ is,

$$\widehat{Var}(\mu_y(\boldsymbol{x})) = (\sigma^2)^{-1} + \boldsymbol{g}^T \boldsymbol{A}^{-1} \boldsymbol{g} \tag{C.9}$$

[3] MacKay, D.J.C. 1991. *"Bayesian Methods for Adaptive Models"*, PhD thesis, California Institute of Technology.

where σ^2 is the (constant) noise variance and A^{-1} is the Hessian matrix of second order partial derivatives (with respect to each of the elements of $\boldsymbol{\Omega}$) of the regularised cost function;

$$\frac{\sigma^2}{2}\sum_{i=1}^{N}\left[\hat{\mu}_y(\boldsymbol{x}_i;\boldsymbol{\Omega})-y_i\right]^2+\frac{\lambda}{2}\sum_i w_i \tag{C.10}$$

The second term in (C.10) is a regularisation term resulting from the assumption that $P(\boldsymbol{\Omega})$, the prior distribution in (B.7) is a Gaussian. In (C.10), λ is a constant and the w_i are the weights from equation (3.10). It follows that an approximate (1-α)100% Bayesian prediction interval for $\mu_y(\boldsymbol{x})$ is given by;

$$\hat{\mu}_y(\boldsymbol{x}_0;\boldsymbol{\Omega}_{MP})\pm z_{(1-\alpha)100\%}\sqrt{(\sigma^2)^{-1}+\boldsymbol{g}_0{}^T\boldsymbol{A}^{-1}\boldsymbol{g}_0} \tag{C.11}$$

By using (C.7), maximum likelihood estimation has been avoided. However, the derivation relies on the same assumptions of normality of the errors and unbiasedness as the delta method, to which it is related. Bishop and Qazaz[4] have extended the method to the case of non-constant variance, replacing the constant noise variance term in (C.9) with input-dependant (variable) noise variance. An advantage of the Bayesian approach is that the regularisation parameter is automatically determined during training. This means cross validation is not required to control over fitting, so all of the available data can be used for training. Unfortunately, for neural nets obtaining Bayesian standard error estimates is substantially more complex than using maximum likelihood based approaches[5]. This is due to the approximations required to obtain the analytical formulae. The Bayesian method is unreliable where crude approximations are used. Moreover, inversion of the Hessian matrix is required, with the attendant possibility of failure.

[4] Bishop, C.M., Qazaz, C.S. 1995. *"Bayesian Inference of Noise Levels in Regression"*, Technical Report, Neural Computing Research Group, Aston University.

[5] Ungar, L.H., De Veaux, R.D., Rosengarten, E. 1995. *"Estimating Prediction Intervals for Artificial neural Networks"*, Department of Computer and information Science, University of Pennsylvania, Philadelphia, PA 19104.

APPENDIX D: Option Pricing

D 1.1 Option Valuation in Continuous Time

Option valuation models can be classified into two categories. Those derived in continuous time, and those derived in discrete time. The continuous time approach is considered first.

The pricing methodology of Black and Scholes (1973) is the foundation of modern option pricing. The essential insight of Black and Scholes was that, under certain assumptions, a risk free hedging portfolio could be constructed which instantaneously replicates the option with certainty. The authors showed that such a portfolio, consisting of a long stock position and a short position in an option could be constructed, assuming that time and the stock price are continuous variables, and that the stock price follows a geometric Brownian motion. By using a technique known as 'dynamic hedging' where the long position in the stock is continuously adjusted, a perfect hedge is maintained, and a unique price for the option obtained. Any option price different to the value of the replicating portfolio gives rise to an arbitrage opportunity. A central assumption underlying this analysis is that there are no transaction costs involved in trading the underlying asset, and that it has a single unique price. If there are transaction costs, however small, then dynamic hedging becomes unworkable. This is because the price process of the underlying asset is one of infinite variation, requiring the hedge portfolio to be rebalanced at every instant, so that the cost of hedging and hence the price of the option, becomes infinite.

Using stochastic calculus the authors derive the stochastic differential equation (s.d.e.) (D.1) for the value f of the option under the above assumptions. (The authors also present an alternative derivation of (D.1) using the capital asset pricing model.).

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf \qquad \text{(D.1)}$$

Since (D.1) contains no terms that are sensitive to investor risk-preference, it is possible to assume investors are risk-neutral. Given the assumption of risk-neutrality, and the boundary conditions imposed by the option payoff formula, there is only one solution to (D.1). The s.d.e. (D.1) was recognized to be identical to the heat transfer equation, and

the required solution was known. The authors used the solution presented in Churchill (1963, p.155), following minor algebraic manipulation, the following formula for the theoretical fair value of a European call, or American call on a dividend-free stock was obtained;

$$C(S,t) = SN(d_1) - Xe^{r(t-T)}N(d_2) \tag{D.2}$$

For a European put the formula is;

$$P(S,t) = Xe^{r(t-T)}N(-d_2) - SN(-d_1) \tag{D.3}$$

There is no corresponding analytical solution in the case of an American put or American call on a dividend paying stock.

In Equations (D.2) and (D.3);

$$d_1 = \frac{\ln(S / Xe^{-r(T-t)}) + \sigma^2(T-t)/2}{\sigma\sqrt{(T-t)}} \tag{D.4}$$

$$d^2 = d^1 - \sigma\sqrt{(T-t)} \tag{D.5}$$

And;

$$e^{r(t-T)} \equiv e^{-r(T-t)} \tag{D.6}$$

Where;

C	=	price of a call option on an underlying asset.
P	=	price of a put option on an underlying asset.
S	=	underlying asset price.
X	=	exercise (strike) price of the option.
r	=	continuously compounded risk free interest rate.
t	=	current date.
T	=	expiration date of the option.
σ	=	the standard deviation of the instantaneous, annualized rate of return on the underlying asset ('volatility').
$N(\bullet)$	=	the cumulative standard normal density function.

The models rest on the following assumptions:

A) The price of the underlying stock follows a lognormal distribution. The stock price moves randomly but smoothly, with no jumps, (i.e. in order to move from one price to another, the underlying stock is assumed to pass through all intermediate prices).

B) The underlying stock pays no dividends. (This assumption may be relaxed.)

Option Valuation in Continuous Time

C) The risk free rate r, and variance σ^2, of returns on the underlying stock are constant for the life of the option.

D) There are no taxes or transaction costs. All securities are perfectly divisible.

Heuristically, theoretical fair value of the (call) option premium is equal to the stock price multiplied by the inverse of the hedge ratio, minus the discounted exercise price multiplied by the probability of exercise, where for each share of stock in the risk-free hedge portfolio, there are $1/N(d_1)$ calls written on the stock. Also, the probability of the option expiring in-the-money, and hence being exercised, is given by $N(d_2)$.

A number of authors have extended the BS option valuation model. These extensions mainly involve relaxation of the assumptions of the model, or adaptations to special cases. Because stock index options, (including the LIFFE FT-SE 100 Index Option) are based not on an underlying stock, but on a futures contract based on the underlying stock the appropriate valuation formula is the Black I formula [Black (1976)]. This was developed by Fischer Black to price European options on commodity futures. The Black model for a (call) option on a future is;

$$C(f,t) = e^{-r(T-t)}[fN(d_1) - XN(d_2)] \tag{D.7}$$

where

$$d_1 = \frac{Ln(f/X) + (\sigma^2/2)(T-t)}{\sigma\sqrt{(T-t)}}$$

and

$$d_2 = d_1 - \sigma\sqrt{(T-t)}$$

In the Black formula, σ is the volatility of the future, f is the future's value. The remaining variables are the same as in the BS model. However, this formula gives the same price as the BS model on the underlying stock. This is because, in the Black formula $N(d_1)$ is multiplied by $fe^{-r(T-t)}$, while in the BS formula it is multiplied by S. However, with no dividends on the underlying asset the futures price is $Se^{r(T-t)}$. Thus, if $se^{r(T-t)}$ is substituted for f in the Black formula, the formula will be the same as the BS formula. This may not be obvious in d_1. However if $Se^{r(T-t)}$ is substituted for f in d_1;

Option Valuation in Continuous Time

$$d_1 = \frac{Ln(Se^{r(T-t)} / X) + (\sigma^2 / 2)(T-t)}{\sigma\sqrt{(T-t)}} \tag{D.8}$$

The expression $Ln(Se^{r(T-t)} / X)$ is equivalent to $Ln(S) + Ln(e^{r(T-t)}) - Ln(X)$. Thus, $Ln(e^{r(T-t)}) = r(T-t)$ so d_1 becomes;

$$d_1 = \frac{Ln(S / X) + (r + \sigma^2 / 2)(T-t)}{\sigma\sqrt{(T-t)}} \tag{D.9}$$

which is d_1 from the BS formula.

The Black formula rests on the same assumptions as the BS formula from which it is derived. There is one important additional assumption however, the option and the future are assumed to expire simultaneously.

D 1.2 Option Valuation in Discrete Time

An alternative approach to option valuation is lattice based models of the type developed by Cox, Ross and Rubinstein (1979). The Cox-Ross-Rubinstein Binomial Model is a discrete model. The price of the underlying stock is assumed to take one of two states, up or down (possibly at different rates) at the end of each period in a single-period binomial model. It is immediately obvious that in a real stock market situation the range of possible outcomes is much greater than the two states a binomial model allows. However, the model can accommodate this by increasing the number of periods. If the life of an option is divided into n periods, as n increases the length of each period will decrease. Thus, if n is made large enough, the discrete binomial model can be made to approximate a continuous model. Under the Cox-Ross-Rubinstein n period Binomial Model, the price of a European call with j periods remaining to expiration is given as;

$$C = \frac{\sum_{j=0}^{n} \frac{n!}{j!(n-j)!} p^j (1-p)^{n-j} MAX[0, S(1+u)^j (1+d)^{n-j} - X]}{(1+r)^n} \tag{D.10}$$

In (D.10);

j = number of periods remaining to expiration.

n = total number of periods.

$u = e^{\sigma\sqrt{(T-t)/n}} - 1$ = up parameter.

Option Valuation in Discrete Time

$$d = \left[\frac{1}{1+u}\right] - 1 = \text{down parameter.}$$

To operate in the binomial framework, the continuously compounded risk-free rate must be converted to a discrete rate r, thus;

$$(1-r)^{(T-t)/n} - 1 = \text{Risk-free rate } r$$

Other variables are as in the BS formula given above. The n period binomial model lends itself well to computer solution. The formula works because the factorial term $n!/j!(n-j)!$ completely enumerates all possible routes a stock price can take to end up at a given level.

The Cox-Ross-Rubinstein binomial model was developed after the BS model. However, the authors show the BS formula is the limiting case of the n period binomial model as $n \to \infty$. Discrete-time lattice based models of the Binomial tree type are more flexible than models of the Black-Scholes type. They are easily adapted to value American put options, barrier options, and various types of exotic options. Most recent advances in option valuation feature models of this type.

REFERENCES

Black, F., and Scholes, M. 1973. "*The Pricing of options and corporate liabilities*", Journal of Political Economy 3, pp 637-654.

Black, F. 1976. "*The Pricing of Commodity Contracts*", Journal of Financial Economics 3, pp 167-179.

Churchill, R.V. 1963. *Fourier Series and Boundary Value Problems*, 2nd ed. New York, McGraw-Hill, p 155.

Cox, J.C., Ross, S.A., Rubinstein, M. 1979. "*Option Pricing: A Simplified Approach*", Journal of Financial Economics 7, pp 229-263.

Fitzgerald, M. D. 1987. "*Financial Options*", Euromoney Publications (1987)

Hull, J. 1997. *"Options, Futures and Other Derivative Securities"*, 3rd. Ed. Prentice Hall International.

Merton, R.C. 1973. *'Theory of Rational Option Pricing.'* Bell Journal of Economic and Management Science 4, pp 141-183.

Rendleman, R.J. and Barttr, B.J. 1979. "*Two-State Option Pricing*", Journal of Finance, 34, pp 1093-1110.

APPENDIX E: Summary Statistics of RNDs

One Month Maturity LIFFE FTSE 100 Index Put Options (Apr. 1992 to Mar. 1997)													
Dates		European Exercise (ESX)						American Exercise (SEI)					
Trading Day	Expiry Month	Median	Ann. IV%	Mean	Std. Dev.	Skew	Ex. Kurt.	Median	Ann. IV%	Mean	Std. Dev.	Skew	Ex. Kurt.
24-Mar-92	Apr-92	2313.57	25.55	2363.63	161.06	1.596	4.55	2301.01	15.93	2325.07	98.78	1.433	4.88
21-Apr-92	May-92	2685.71	16.12	2645.07	115.70	-0.648	2.76	2670.42	16.69	2627.76	119.10	-0.699	3.07
27-May-92	Jun-92	2742.16	12.47	2707.48	91.37	-0.577	3.08	2742.12	12.16	2700.37	89.13	-0.947	3.72
24-Jun-92	Jul-92	2584.60	14.66	2555.61	101.25	0.163	3.42	2574.03	14.27	2540.95	98.20	0.076	3.29
29-Jul-92	Aug-92	2463.65	16.26	2432.90	107.08	-0.227	2.39	2461.09	15.56	2422.47	102.35	-0.479	2.38
25-Aug-92	Sep-92	2315.42	17.98	2296.90	111.24	0.356	2.24	2309.69	16.18	2284.14	99.88	0.090	1.76
23-Sep-92	Oct-92	2626.74	16.68	2592.60	117.11	-0.475	2.62	2607.95	15.88	2566.19	110.68	-0.727	3.17
28-Oct-92	Nov-92	2689.30	16.29	2652.84	117.08	-0.546	2.66	2681.81	15.62	2640.45	111.94	-0.714	3.18
25-Nov-92	Dec-92	2750.36	15.20	2716.88	111.71	-0.421	2.76	2743.10	13.95	2701.34	102.27	-0.788	3.39
21-Dec-92	Jan-93	2854.45	16.94	2820.62	129.26	-0.415	2.52	2835.39	15.54	2793.51	117.76	-0.698	3.08
27-Jan-93	Feb-93	2883.43	13.26	2848.44	102.21	-0.539	2.91	2869.01	12.54	2827.84	96.14	-0.855	3.56
24-Feb-93	Mar-93	2847.93	14.13	2813.25	107.57	-0.479	2.87	2852.12	13.33	2810.21	101.58	-0.796	3.44
22-Mar-93	Apr-93	2886.20	14.95	2852.32	115.35	-0.418	2.72	2897.45	13.95	2855.48	108.04	-0.735	3.16
27-Apr-93	May-93	2866.78	13.12	2831.73	100.53	-0.527	2.96	2865.82	12.26	2824.74	93.87	-0.859	3.60
25-May-93	Jun-93	2882.31	13.43	2847.23	103.43	-0.537	2.90	2874.80	12.63	2833.15	97.07	-0.874	3.55
23-Jun-93	Jul-93	2948.43	11.88	2912.37	93.58	-0.604	3.06	2942.98	11.07	2901.22	87.08	-0.964	3.78
28-Jul-93	Aug-93	2931.82	11.69	2893.49	91.56	-0.757	3.13	2924.84	11.54	2884.33	90.18	-0.887	3.65
24-Aug-93	Sep-93	3097.19	11.83	3060.73	97.92	-0.603	3.00	3090.45	11.19	3047.91	92.42	-0.940	3.67
22-Sep-93	Oct-93	3045.93	12.59	3009.68	102.46	-0.576	2.92	3044.72	12.21	3003.08	99.35	-0.802	3.44
27-Oct-93	Nov-93	3200.41	11.20	3163.45	95.83	-0.624	3.05	3194.63	10.80	3152.31	92.21	-0.909	3.65
24-Nov-93	Dec-93	3100.00	15.08	3066.31	124.90	-0.391	2.71	3094.02	14.76	3054.99	122.05	-0.591	3.03
23-Dec-93	Jan-94	3438.04	11.52	3391.64	105.84	-0.855	3.11	3432.76	11.87	3386.07	108.89	-0.897	3.32
26-Jan-94	Feb-94	3473.58	12.98	3438.71	120.54	-0.436	2.72	3465.37	12.84	3426.63	118.95	-0.594	2.98
23-Feb-94	Mar-94	3366.86	14.25	3333.39	128.25	-0.367	2.63	3363.51	14.24	3327.00	128.05	-0.480	2.83
21-Mar-94	Apr-94	3215.27	16.33	3184.63	140.31	-0.208	2.37	3218.22	15.84	3183.61	136.27	-0.337	2.43
26-Apr-94	May-94	3164.31	14.42	3130.87	121.99	-0.304	2.38	3153.92	14.09	3116.26	118.73	-0.409	2.38
24-May-94	Jun-94	3110.40	13.83	3074.88	114.94	-0.491	2.82	3123.59	13.76	3083.76	114.85	-0.659	3.13
22-Jun-94	Jul-94	2990.44	16.53	2959.58	132.13	-0.215	2.34	2981.40	15.97	2945.77	127.28	-0.341	2.33
27-Jul-94	Aug-94	3112.18	13.67	3076.87	113.72	-0.480	2.76	3117.40	13.53	3077.43	112.73	-0.648	3.04
23-Aug-94	Sep-94	3224.98	12.97	3189.00	111.80	-0.524	2.80	3209.36	12.80	3168.20	109.80	-0.714	3.17
28-Sep-94	Oct-94	3073.19	15.53	3040.65	127.52	-0.337	2.55	3066.40	14.69	3026.89	120.41	-0.576	2.80
26-Oct-94	Nov-94	3018.80	16.47	2988.00	132.89	-0.237	2.45	3028.95	15.27	2990.12	123.58	-0.511	2.66
23-Nov-94	Dec-94	3051.35	19.38	3027.62	158.09	-0.006	2.43	3047.91	17.01	3011.78	138.60	-0.373	2.51
23-Dec-94	Jan-95	3133.58	14.31	3098.97	119.85	-0.429	2.72	3117.10	13.31	3075.13	110.89	-0.736	3.22
25-Jan-95	Feb-95	3024.50	14.36	2990.82	116.06	-0.411	2.72	3012.90	13.21	2970.71	106.36	-0.780	3.22
22-Feb-95	Mar-95	3050.37	13.04	3015.06	106.33	-0.499	2.86	3056.38	12.26	3014.72	100.13	-0.800	3.40
27-Mar-95	Apr-95	3195.50	13.26	3160.52	113.28	-0.462	2.79	3181.16	12.42	3139.75	105.59	-0.762	3.32
25-Apr-95	May-95	3254.46	11.74	3217.96	102.12	-0.572	2.95	3251.28	11.14	3209.07	96.84	-0.833	3.53
23-May-95	Jun-95	3340.38	11.70	3303.93	104.42	-0.578	2.91	3326.13	11.09	3284.11	98.55	-0.870	3.48
28-Jun-95	Jul-95	3307.80	12.53	3272.63	110.80	-0.480	2.81	3307.85	11.71	3266.14	103.49	-0.784	3.32
26-Jul-95	Aug-95	3486.92	11.39	3450.60	106.14	-0.565	2.88	3487.42	10.87	3446.15	101.30	-0.787	3.39
22-Aug-95	Sep-95	3586.42	10.69	3549.80	102.51	-0.588	2.97	3565.49	10.24	3524.28	97.56	-0.832	3.48
27-Sep-95	Oct-95	3509.83	11.62	3473.92	108.95	-0.512	2.86	3515.70	10.90	3474.01	102.38	-0.811	3.35
25-Oct-95	Nov-95	3580.64	11.75	3544.96	112.41	-0.504	2.79	3570.99	11.09	3529.87	105.85	-0.751	3.27
22-Nov-95	Dec-95	3677.14	11.29	3640.96	110.94	-0.533	2.81	3663.20	10.67	3621.49	104.42	-0.792	3.29
22-Dec-95	Jan-96	3690.57	10.79	3652.79	106.44	-0.607	2.93	3699.25	10.37	3656.97	102.52	-0.790	3.39
24-Jan-96	Feb-96	3808.51	10.42	3771.13	106.02	-0.577	2.94	3789.99	10.19	3749.31	103.19	-0.747	3.31
21-Feb-96	Mar-96	3758.83	10.80	3721.93	108.54	-0.544	2.89	3758.68	10.43	3717.36	104.74	-0.767	3.26
25-Mar-96	Apr-96	3700.69	11.91	3664.42	117.81	-0.478	2.75	3713.84	11.51	3673.80	114.27	-0.645	3.07
23-Apr-96	May-96	3881.05	10.47	3843.78	108.64	-0.576	2.89	3870.91	10.29	3830.32	106.43	-0.725	3.24
29-May-96	Jun-96	3827.20	10.09	3790.05	103.22	-0.596	2.99	3815.47	9.85	3774.34	100.44	-0.806	3.37
26-Jun-96	Jul-96	3725.59	10.72	3688.77	106.77	-0.563	2.90	3732.39	10.60	3692.23	105.79	-0.709	3.20
24-Jul-96	Aug-96	3694.95	12.16	3659.70	120.09	-0.439	2.67	3697.07	11.91	3658.19	117.72	-0.569	2.89
28-Aug-96	Sep-96	3987.53	9.79	3949.79	104.31	-0.613	2.98	3962.04	9.70	3921.00	102.71	-0.779	3.30
25-Sep-96	Oct-96	3991.94	10.13	3953.90	108.11	-0.606	2.90	3974.33	10.15	3934.51	107.80	-0.688	3.18
23-Oct-96	Nov-96	4072.83	10.30	4035.74	112.15	-0.547	2.84	4067.74	10.18	4028.21	110.63	-0.655	3.11
27-Nov-96	Dec-96	4105.62	10.00	4068.02	109.71	-0.558	2.89	4086.24	9.74	4046.17	106.36	-0.707	3.22
23-Dec-96	Jan-97	4128.29	10.13	4090.65	111.79	-0.570	2.86	4124.17	9.84	4083.87	108.42	-0.700	3.19
29-Jan-97	Feb-97	4247.83	11.87	4215.61	134.77	-0.312	2.530	4230.74	11.29	4195.17	127.69	-0.456	2.766
26-Feb-97	Mar-97	4355.22	11.19	4321.27	130.25	-0.367	2.59	4356.85	10.82	4320.78	126.03	-0.470	2.79

APPENDIX F: Statistical Tests

A number of statistical tests are employed and discussed, in this thesis. These tests are briefly summarised here:

The t-Test.

Student's t-test for independent samples is used to determine whether two samples are from populations with different means. If both samples are large, the *separate* or *unequal variances* versions of the t-test are appropriate. The Central Limit Theorem guarantees the validity of the test even for populations with non-normal distributions. Sample sizes as small as 30 per group are acceptable if the two populations are approximately normally distributed. The more the populations depart from normality, the larger the sample size needed. However, 100 observations per group is often quite sufficient. For small and moderate sample sizes, the *equal variances* version of the test is an exact test of the equality of two population means. The equal variances t-test requires samples from normally distributed populations with equal (population) standard deviations for valid results. Table 1 gives test statistics and confidence intervals for the unequal variances version of the test where standard deviations are known, the unequal variances version where standard deviations are unknown, and the equal variances version.

Table 1 Two-sample t-tests for Unpaired Data

Assumptions	Confidence Interval	Hypothesis Test	Test Statistic	Decision Rule
Any population, σ known, n_1, n_2 large (≥30)	$(\bar{x}_1-\bar{x}_2)\pm z_{\alpha/2}\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}$	(i) H_0: $\mu_1-\mu_2=0$, H_A: $\mu_1-\mu_2\neq 0$ (ii) H_0: $\mu_1-\mu_2=0$, H_A: $\mu_1-\mu_2>0$ (iii) H_0: $\mu_1-\mu_2=0$, H_A: $\mu_1-\mu_2<0$	$z=\frac{\bar{x}_1-\bar{x}_2}{\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}}$	(i) $z>z_{\alpha/2}$ or $z<-z_{\alpha/2}$ (ii) $z>z_\alpha$ (iii) $z<-z_\alpha$
Any population, σ unknown, n_1, n_2 large (≥30)	$(\bar{x}_1-\bar{x}_2)\pm z_{\alpha/2}\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}$	(i) H_0: $\mu_1-\mu_2=0$, H_A: $\mu_1-\mu_2\neq 0$ (ii) H_0: $\mu_1-\mu_2=0$, H_A: $\mu_1-\mu_2>0$ (iii) H_0: $\mu_1-\mu_2=0$, H_A: $\mu_1-\mu_2<0$	$z=\frac{\bar{x}_1-\bar{x}_2}{\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}}$	(i) $z>z_{\alpha/2}$ or $z<-z_{\alpha/2}$ (ii) $z>z_\alpha$ (iii) $z<-z_\alpha$
Population normal, σ unknown, n_1, and n_2 small (each<30) df=min(n_1-1, n_2-1)	$(\bar{x}_1-\bar{x}_2)\pm t_{\alpha/2}\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}$	(i) H_0: $\mu_1-\mu_2=0$, H_A: $\mu_1-\mu_2\neq 0$ (ii) H_0: $\mu_1-\mu_2=0$, H_A: $\mu_1-\mu_2>0$ (iii) H_0: $\mu_1-\mu_2=0$, H_A: $\mu_1-\mu_2<0$	$t=\frac{\bar{x}_1-\bar{x}_2}{\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}}$	(i) $t>t_{\alpha/2}$ or $t<-t_{\alpha/2}$ (ii) $t>t_\alpha$ (iii) $t<-t_\alpha$

Where the population variance is unknown, it is estimated from the corresponding sample using:

$$s^2=\frac{\sum x^2-\frac{\left(\sum x\right)^2}{n}}{n-1} \qquad \text{(F.1)}$$

A paired t-test, or *t-test for two correlated samples*, is used to determine whether there is a significant difference between the average values of the same measurement made under two different conditions. Both measurements are made on each unit in a sample, and the test is based on the paired differences between these two values. The usual null hypothesis is that the difference in the mean values is zero. The paired sample t-test is a more powerful alternative to a two-sample t-test, but can only be used when there are matched samples. Pairing is usually optional. In most cases a paired analysis or one that uses independent samples can be designed. Table 2 gives the test statistic and confidence intervals for the paired t-test.

Table 2 Two-sample t-test for Paired Data

Assumptions	Confidence Interval	Hypothesis Test	Test Statistic	Decision Rule
Matched pairs of observations for the same n units. Difference $d = x_{con1} - x_{con2}$.	$\bar{d} \pm t_{\alpha/2}\sqrt{\frac{n\sum d^2 - (\sum d)^2}{n^2(n-1)}}$	(i) $H_0: \bar{d} = 0, H_A: \bar{d} \neq 0$ (ii) $H_0: \bar{d} = 0, H_A: \bar{d} > 0$ (iii) $H_0: \bar{d} = 0, H_A: \bar{d} < 0$	$t = \frac{\bar{d}}{\sqrt{\frac{n\sum d^2 - (\sum d)^2}{n^2(n-1)}}}$	(i) $t > t_{\alpha/2}$ or $t < -t_{\alpha/2}$ (ii) $t > t_\alpha$ (iii) $t < -t_\alpha$

The null hypothesis for a two-sample paired t-test is H_0: $\bar{d} = \bar{x}_1 - \bar{x}_2 = 0$ where $\bar{d}$ is the mean value of the difference. This is tested against one of the following alternative hypotheses, depending on the question posed: H_A: $\bar{d} \neq 0$, H_A: $\bar{d} > 0$, H_A: $\bar{d} < 0$.

The t-test is also used to perform tests on the parameter values in regression models. Table 3 shows how the test is applied in the case of multivariate linear regression. The t-test for univariate linear regression, is a special case of the version for multivariate linear regression.

Table 3 Multivariate Linear Regression: t-tests for j^{th} Gradient Parameter

Assumptions	Confidence Interval	Hypothesis Test	Test Statistic	Decision Rule
Normality, independence , and constant variance , of the errors e.	$\beta_j \pm t_{\alpha/2}\sqrt{\frac{\sum e_i^2}{n-k}x_{jj}}$ (n-k) degrees of freedom	(i) $H_0: \beta_j = 0, H_A: \beta_j \neq 0$ (ii) $H_0: \beta_j = 0, H_A: \beta_j > 0$ (iii) $H_0: \beta_j = 0, H_A: \beta_j < 0$	$t = \frac{\hat{\beta}_j - \beta_j}{\sqrt{\frac{\sum e_i^2}{n-k}x_{jj}}}$ x_{jj} is the jth diagonal element of $(x'x)^{-1}$	(i) $t > t_{\alpha/2}$ or $t < -t_{\alpha/2}$ (ii) $t > t_\alpha$ (iii) $t < -t_\alpha$

The F test.

The F test is a test to determine whether two samples are from populations with different variances. The test statistic is given by:

$$F_{n_1-1,n_2-1} = \frac{s_1^2}{s_2^2}, \text{ where } s_1^2 \text{ is larger than } s_2^2. \qquad \text{(F.2)}$$

s_1^2 and s_2^2 are the sample variances. The null hypothesis is H_0: $\sigma_1^2 - \sigma_2^2 = 0$. This is tested against one of the following alternative hypotheses, depending on the question posed: H_A: $\sigma_1^2 - \sigma_2^2 \neq 0$, H_A: $\sigma_1^2 - \sigma_2^2 > 0$, H_A: $\sigma_1^2 - \sigma_2^2 < 0$.

White's test

This is a test for heteroskedasticity in the residuals from a least squares regression. White's test is a test of the null hypothesis of no heteroskedasticity against heteroskedasticity of some unknown general form. The test statistic is computed by an auxiliary regression, where the squared residuals are regressed on all possible (non-redundant) cross products of the explanatory variables. White's test statistic, is computed as:

$$W = nR^2 \qquad \text{(F.3)}$$

from the auxiliary regression. In (F.3) n is the number of observations. White's test statistic is asymptotically χ^2 distributed, with degrees of freedom equal to the number of slope coefficients (excluding the constant) in the auxiliary regression. White considers this approach as a general test for model mis-specification, since the null hypothesis underlying the test assumes that the errors are both homoskedastic and independent of the regressors, and that the linear specification of the model is correct. Failure of any one of these conditions could lead to a significant test statistic. Conversely, a non-significant test statistic implies that none of the three conditions is violated.

The Jarque-Bera Test

The Jarque-Bera test is for testing whether a series is normally distributed. The test statistic measures the difference between the skewness and kurtosis of the series and those of a normal distribution. The test statistic is computed as:

$$JB = \frac{n-k}{6}\left(S^2 + \frac{1}{4}(K-3)^2\right) \tag{F.4}$$

in (F.4) S is the skewness, K is the kurtosis, n is the number of observations in the series, and k is the number of estimated coefficients (if any) used to create the series. Under the null hypothesis of a normal distribution, the Jarque-Bera statistic is χ^2 distributed with 2 degrees of freedom. The associated P value is the probability that a Jarque-Bera statistic exceeds (in absolute value) the observed value under the null hypothesis, a small probability value leads to the rejection of the null hypothesis of a normal distribution.

The LM test

The LM (Lagrange multiplier) test can be used to test parameter restrictions by estimating *only the restricted equation.* The LM test statistic can be computed as;

$$LM = nR^2 \tag{F.5}$$

from an auxiliary regression[1]. The LM test statistic is χ^2 distributed with degrees of freedom equal to the number of restrictions imposed.

The LR test

This test is used to add extra explanatory variables to an existing equation and to ask whether they make a significant contribution to explaining the variation in the dependent variable. The null hypothesis is that the additional regressors are not jointly significant. The output from the test is a likelihood ratio (LR) statistic with associated p-values, or an F-statistic, together with the estimation results for the *unrestricted model* under the alternative hypothesis. The F-statistic is based on the difference between the residual sums of squares of the restricted and unrestricted regressions, *thus both must be estimated.* The LR statistic is computed as;

$$LR = -2(L^U - L^R) \tag{F.6}$$

where L^U and L^R are the values of the maximized log likelihood function for the unrestricted and restricted regressions, respectively. Under the null hypothesis of no significance for the additional regressors, the LR statistic has an asymptotic χ^2 distribution with degrees of freedom equal to the number of restrictions, i.e. the number of added variables.

[1] Details of the LM, LR, and Wald tests can be found in Thomas, R.L. 1993. *Introductory Econometrics: Theory and Applications*, 2nd edn., Harlow: Longman. Chapter 4.

The Wald Test.

The Wald test can be used to test parameter restrictions by estimating *only the unrestricted equation*. It computes the test statistic by estimating the unrestricted regression without imposing the coefficient restrictions specified by the null hypothesis. The Wald statistic measures how close the unrestricted estimates come to satisfying the restrictions under the null hypothesis. If the restrictions are in fact true, then the unrestricted estimates should come close to satisfying the restrictions. For a nonlinear regression

$$y = (\beta)\boldsymbol{X} + \varepsilon \tag{F.7}$$

where β is a vector of k parameters to be estimated, restrictions on the parameters can be written as $H_0 : g(\beta) = 0$. Here, g is a smooth q dimensional vector imposing q restrictions on β. The Wald statistic is given by;

$$W = ng(\boldsymbol{b})'\left(\frac{\partial g}{\partial \beta} V \frac{\partial g}{\partial \beta'}\right), \text{ where } V = ns^2\left(\frac{\partial x}{\partial \beta}\frac{\partial x}{\partial \beta'}\right), s^2 = \frac{\boldsymbol{u}'\boldsymbol{u}}{n-k} \tag{F.8}$$

In (F.8) n is the number of observations, $\boldsymbol{b}$ is the unrestricted parameter estimates, V is the estimated variance of b, and $\boldsymbol{u}$ are the residuals from the unrestricted model. Under the null hypothesis H_0, the Wald statistic has an asymptotic (q) distribution, where q is the number of restrictions.

Further details of the construction of the LM, LR, and Wald statistics can be found in Thomas (1993) Chapter 4.

APPENDIX G: Material Published in Advance of the Thesis

1) Healy, J.V., Dixon, M., Read, B.J., and Cai, F.F. 2001. *" A Data-Centric Approach to Understanding the Pricing of Financial Options*", European Physics Journal B: Proceedings of the 3rd International Conference on Applications of Physics to Financial Analysis 2001, pp219-227.
2) Healy, J.V., Dixon, M., Read, B.J., and Cai, F.F. 2003. *"Confidence in Data Mining Model Predictions: A Financial Engineering Application"*, 29th Annual Conference of the IEEE Industrial Electronics Society, Special Session on Intelligent Systems, pp1926-1931.
3) Healy, J.V., Dixon, M., Read, B.J., and Cai, F.F. 2003. *"Confidence Limits for Data Mining Models of Options Prices"*, European Physics Journal A: Proceedings of the 4th International Conference on Applications of Physics to Financial Analysis 2003.
4) Xu, L., Dixon, M., Eales, B. A., Cai, F. F., Read, B. J., and Healy, J. V. 2003. *"Barrier Option Pricing: Modelling with Neural Nets"*, European Physics Journal A: Proceedings of the 4th International Conference on Applications of Physics to Financial Analysis 2003.

PAGE/PAGES EXCLUDED UNDER INSTRUCTION FROM THE UNIVERSITY

APPENDIX H: Approved Program of Research

Extracts From the Approved Document
(Form RD/R approved 22 November 2000)

4 The Programme of Research

4.1 Title of the proposed investigation:

Computational Knowledge Discovery Techniques and their Application to Options Market Databases

4.2 Aim of the investigation

The aim is to provide a framework for building and proving a computer system. The framework;

- Determines which computational knowledge discovery techniques are appropriate to financial options databases.
- Derives option pricing models and forecasts of prices and price variability from the discovered data patterns.
- Selects appropriate statistical criteria for the assessment of the confidence that can be placed upon the extracted patterns and models.
- Formulates trading strategies based upon the difference in price probabilities of options and assets.

4.3.1 Introduction

.....There is a need to provide a systems framework in which the criteria and assumptions are visible for the selection of datasets, the choice of relevant time span, the selection of knowledge discovery algorithms, the possible use of data cleaning strategies, the extrapolation to future time steps of data, extraction of a pricing model, the refinement of the findings, the establishment of a trading strategy, and a statistical assessment of the effectiveness of the strategy. The aim of this work is to provide a computational tool, which implements the required systems framework.

4.3.3 Rationale & Proposal

.....In summary the goals of the research are;

- To investigate a specific question of interest in the financial options market domain utilising data mining / KDD techniques.
- To use the investigation as a vehicle for the development of methodologies and new techniques for the effective application of data mining / KDD within this specialist domain, with consideration of their wider applicability.
- To use the investigation to investigate the comparative merits of data mining / KDD approaches to modelling versus current approaches in this domain.

4.3.5 Benefits expected

.....The above agenda incorporates several new applications for data mining / KDD within the specialist domain of options markets. A data mining / KDD based approach offers a number of advantages over current practices for development of pricing models in options markets. First, there is the potential for a high level of automation and, in suitable circumstances full automation, allowing implementation within automated trading systems. Second, it provides an alternative data driven approach to modeling, which is independent of many of the theoretical assumptions that underpin the conventional analytical models and their numerical and analytical approximations.

.....The benefits of the research may be summarized as follows;

For Computing / Information Systems

- Evaluation of the effectiveness and limitations of data mining /KDD for extracting information in the specific context of options market data.
- Development of methodologies and techniques for more effective application of data mining / KDD technology to financial market data in general and derivatives markets in particular.
- Testing of commonly made modelling assumptions and identification of what cannot be done without some model or assumption.
- Identification of those methodologies, techniques and algorithms developed that have potential application in KDD for other data domains.

For Finance

- Improved financial risk management and forecasting.
- Better pricing and hedging of investment portfolios.
- New approaches to discovering trading strategies.

4.3.6 Data Sources

It is expected that the data used will be obtained from the Market Data Services division of LIFFE (London International Financial Futures and Options Exchange. The LIFFE data set will be enriched with additional data obtained from the Datastream database. Substantial datasets of daily closing prices, transaction prices, and relevant interest rates are available from these sources and we already have some of these available.

4.3.7 Software

We are currently using the commercially available SPSS data-mining package Clementine. The SPSS statistics package is being used for the statistical elements of the work. Other software will be deployed/developed as appropriate.

4.3.10 Original Contribution to Knowledge

A systematic methodology for applying computerised knowledge discovery to option pricing and the extraction of forecasts from options market data.

RD/R approved 22.11.2000